**CEIT – 04 – 502A**
**EE04L / MWF / 4:30p – 6:00p**
**LABORATORY REPORT 3**

GROUP 6
Group Members:
Bernardo, Raevon Thaddeus C.
Bertumen, Charles Jefferson
Cabanes, Christine Joy P.
Cesar, John Lester M.
Landicho, Bhaves Nicolette D.
Solis, Johnloyd P.

# Machine Problem 3: Root Approximation by Open Methods

**Program Name:** Group 6 Iterative Open Method Root Approximator

**Acronym:** G6–IOMRA

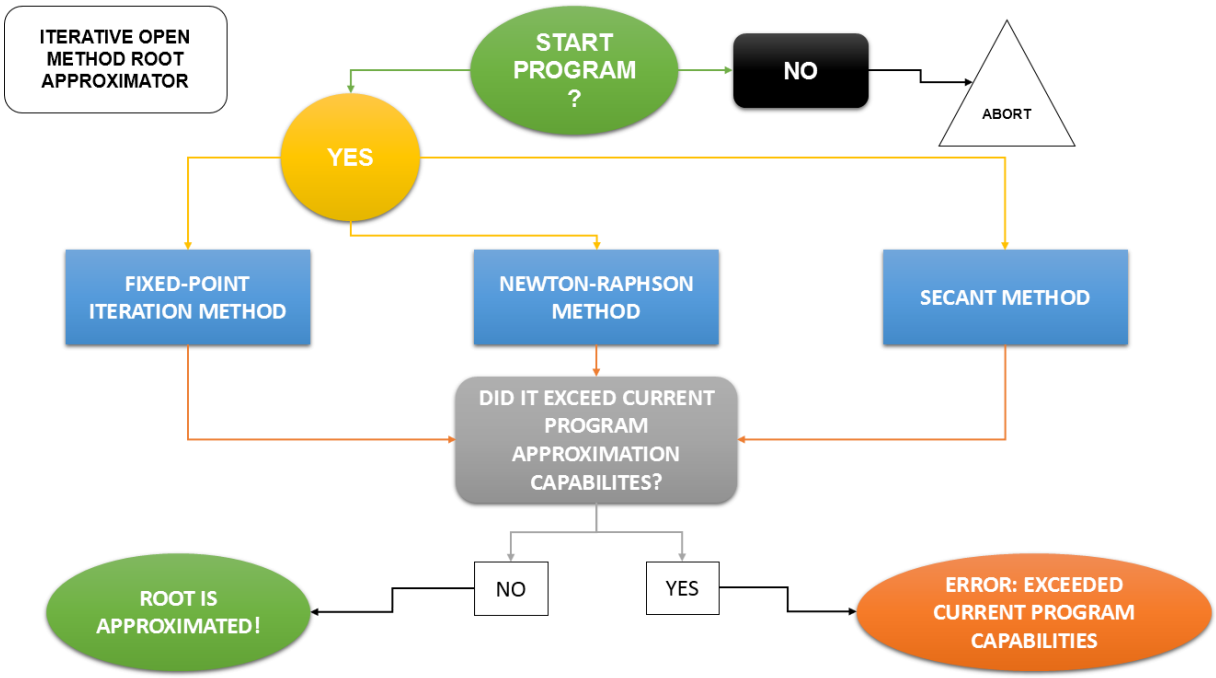
**Current Version:** 1.0.1

**Version History:**

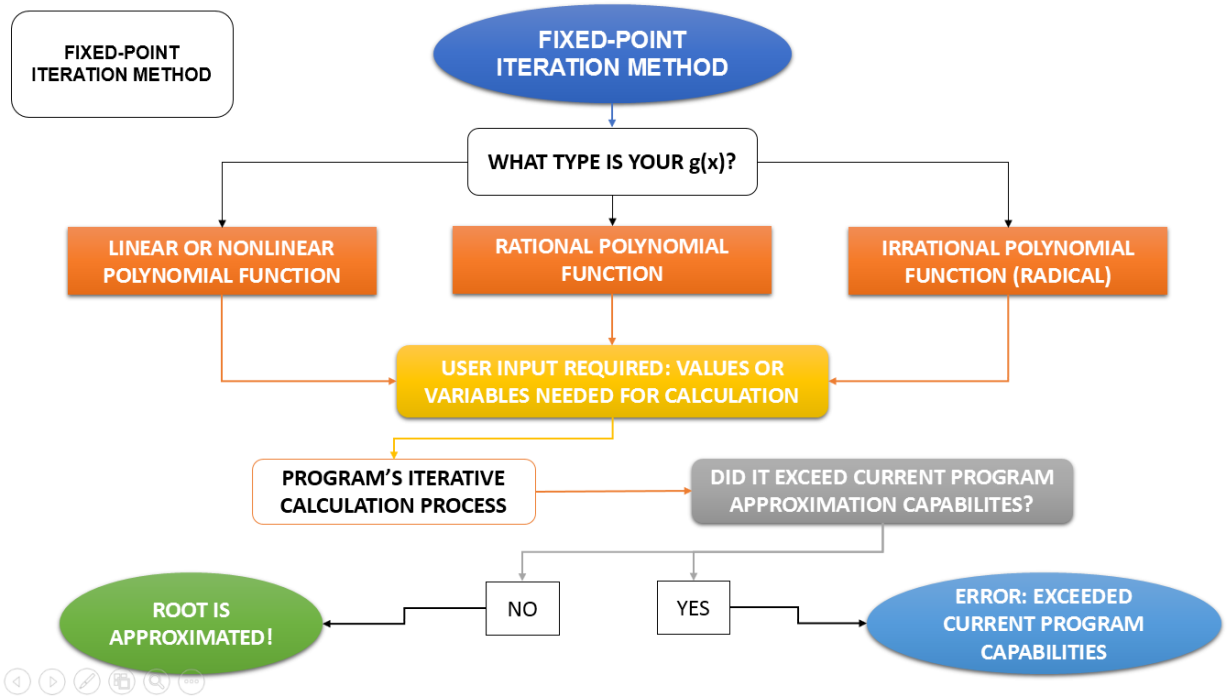
- **1.0.1**
- Trial version
- Designed in Scilab version 6.0.2
- Fixed-Point Iteration Method Mode can only approximate up to 25$^{th}$ approximation
- Newton-Raphson Method Mode can only approximate up to 20$^{th}$ approximation
- Secant Method Mode can only approximate up to 20$^{th}$ approximation

## TABLE OF CONTENTS

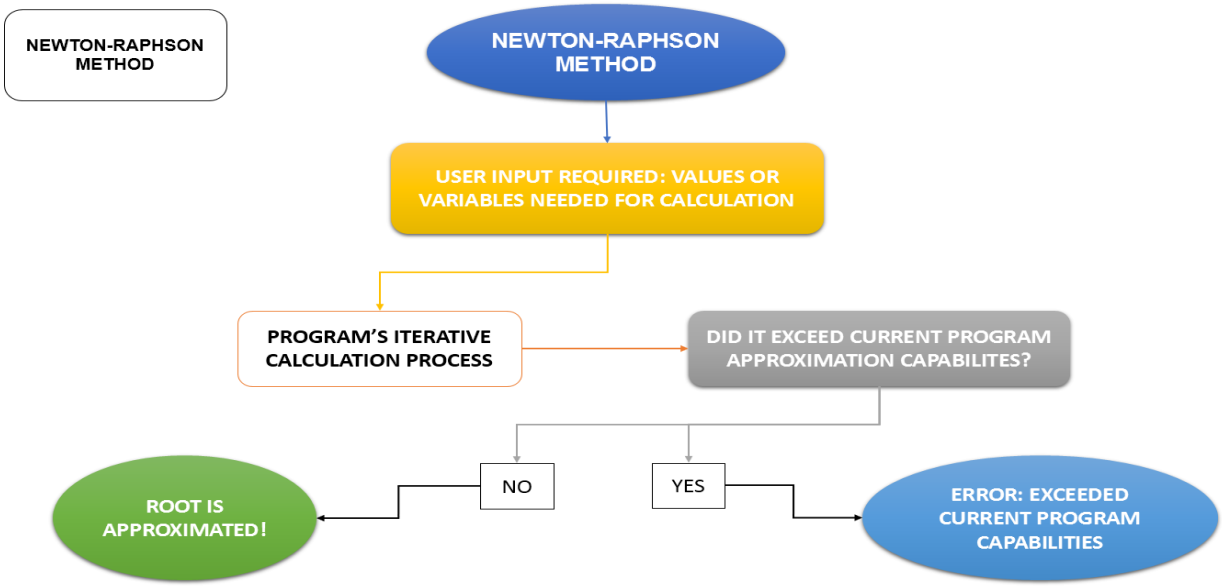
## I. Flow Chart



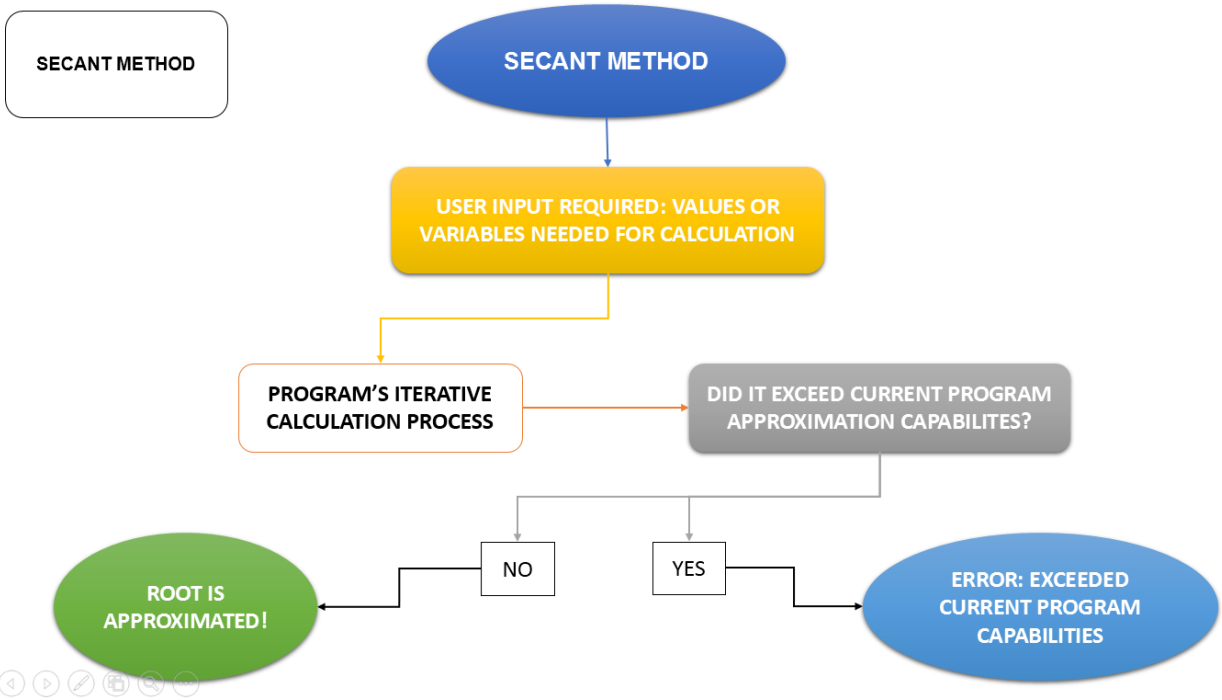

The program starts with an option to proceed or abort. Upon agreeing to start the program, the user will be prompted by the program to choose one from the three modes under Iterative Open Methods: **(a) Fixed-Point Iteration Method**, **(b) Newton-Raphson Method** and **(c) Secant Method**. The gist of the program is that it will absolutely approximate the root that you are solving for, provided that the iteration count or total number of approximations calculated by the program does not exceed their respective program mode capabilities. This is due to the program itself being only a trial version created by Group 6 as a course requirement activity. However, if needed so, its algorithms can further be improved to cater approximations exceeding its current capabilities.




In this mode, the program will inquire what type of **g(x)** will the user be inputting. Currently, it only accepts functions that are **(a) linear or nonlinear polynomial functions**, **(b) rational polynomial functions** and **(c) irrational polynomial functions (radical)**. This embedded mechanics of the program alliows the user to re-write their functions in case their current function format does not converge into its root. The user is then prompted to input values or variables that are needed for the iterative calculation process of the program. As long as calculations do not exceed the 25$^{th}$ approximation, the root will always be determined.



In this mode, the user will be prompted to input values or variables that are needed for the iterative calculation process of the program. More specifically, it will prompt the user to input **f(x)**, **f'(x)** and $\mathbf{x_0}$. As long as calculations do not exceed the 20th approximation, the root will always be determined.




In this mode, the user will be prompted to input values or variables that are needed for the iterative calculation process of the program. More specifically, it will prompt the user to input **f(x)**, $\mathbf{x_0}$ and $\mathbf{x_1}$. As long as calculations do not exceed the 20th approximation, the root will always be determined.

## II. Source Code of the Working Program

```
clc
yes=1
YES=1
Yes=1
no=0
NO=0
No=0
disp("Welcome to Group 6 Iterative Open Method Root Approximator 1.0.1 (G6-IOMRA 1.0.1)!")
disp("Version: 1.0.1 (TRIAL VERSION)")
disp("")
disp("Current Version Capabilities:")
disp("Fixed-Point Iteration Method: Can approximate up to 25th approximation.")
disp("Newton-Raphson Method: Can approximate up to 20th approximation.")
disp("Secant Method: Can approximate up to 20th approximation.")
disp("")
disp("")
ANSWER0=input (" Start program? input Yes to start and No to abort:  ")
if ANSWER0==1 then
   clc
   a=213
   b=321
   c=123
   disp("Please choose desired program mode:")
   disp("(a) Fixed-Point Iteration Method")
   disp("(b) Newton-Raphson Method")
   disp("(c) Secant Method")
   disp("")
   ANSWER1=input (" Your choice:  ")
   if ANSWER1==213 then
      clc
      a=123
      b=213
      c=312
      disp("What type is your g(x)?:")
      disp("(a) Linear or Nonlinear Polynomial Function")
      disp("(b) Rational Polynomial Function")
      disp("(c) Irrational Polynomial Function (Radical)")
      disp("")
      ANSWER2=input (" Your choice:  ")
      if ANSWER2==123 then
         clc
         disp("Mode: Fixed-Point Iteration Method")
         disp("Type: Linear or Nonlinear Polynomial Function")
         disp("How to Use G6-IOMRA 1.0.1: Input polynomial function in this format: poly([a,b,c,--
>nth],%x%,%coeff%)")
         disp("Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desired
polynomial function in an ascending order in terms of degree.")
         disp("")
         disp("(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab
coding mechanics so please replace % with quotation mark!)")
         disp("")
         disp("e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2")
         disp("")
         y=input (" Input your g(x):  ")
```

```
        x=input (" Input your x0:  ")
        z=horner(y,[x])
        z1=z*1000
        z2=ceil(z1)
        x1_rounded=z2/1000
    elseif ANSWER2==213 then
        clc
        disp("Mode: Fixed-Point Iteration Method")
        disp("Type: Rational Polynomial Function")
        disp("How to Use G6-IOMRA 1.0.1: Input polynomial function like how you normally input rational
functions.")
        disp("e.g. (x+2)/(x+3)")
        disp("")
        x=poly(0,"x")
        y=input (" Input your g(x):  ")
        x=input (" Input your x0:  ")
        z=horner(y,[x])
        z1=z*1000
        z2=ceil(z1)
        x1_rounded=z2/1000
    elseif ANSWER2==312 then
        clc
        disp("Mode: Fixed-Point Iteration Method")
        disp("Type: Irrational Polynomial Function (Radical)")
        disp("How to Use G6-IOMRA 1.0.1: Input polynomial function in this format: poly([a,b,c,--
>nth],%x%,%coeff%)")
        disp("Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desired
polynomial function in an ascending order in terms of degree.")
        disp("")
        disp("(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab
coding mechanics so please replace % with quotation mark!)")
        disp("")
        disp("e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2")
        disp("")
        disp("When iputting your polynomial function, dont mind the nthroot. Wait for program nthroot
prompt.")
        disp("You are also required to input the nthroot of your irrational polynomial function (radical)")
        disp("eg. with poly([1,2,3],%x%,%coeff%), and nthroot input of 3:")
        disp("This will result in (1+2x+3x^2)^(1/3)")
        disp("")
        y=input (" Input your g(x):  ")
        z_nthroot=input (" Input the nthroot:  ")
        x=input (" Input your x0:  ")
        z=horner(y,[x])
        x1=nthroot (z,z_nthroot)
        x2=x1*1000
        x3=ceil(x2)
        x1_rounded=x3/1000
    else
        clc
        disp("ERROR ENCOUNTERED!")
        disp("Please contact Head Developer through Email:")
        disp("bernardoraevon@gmail.com")
        abort
    end
    // x0 or initial approxiumation
```

```
disp("x0 or initial approximation:  "+string(x)+"")
// x1 or first approximation
a1=x1_rounded;
disp("x1 or first approximation:  "+string(x1_rounded)+"")
// LINEAR, NONLINEAR, OR RATIONAL POLYNOMIAL ALGORITHM BRANCH
if ANSWER2==123 | ANSWER2==213 then
    a2=horner(y,[a1])
    a3=a2*1000
    a4=ceil(a3)
    x2_rounded=a4/1000
// x2 or second approximation
    disp("x2 or second approximation:  "+string(x2_rounded)+"")
    if x2_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x2_rounded)+"")
        abort
    end
    b1=x2_rounded
    b2=horner(y,[b1])
    b3=b2*1000
    b4=ceil(b3)
    x3_rounded=b4/1000
// x3 or third approximation
    disp("x3 or third approximation:  "+string(x3_rounded)+"")
    if x3_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    elseif x3_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    end
    c1=x3_rounded
    c2=horner(y,[c1])
    c3=c2*1000
    c4=ceil(c3)
    x4_rounded=c4/1000
// x4 or fourth approximation
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"")
    if x4_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    elseif x4_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    end
    d1=x4_rounded
    d2=horner(y,[d1])
    d3=d2*1000
    d4=ceil(d3)
    x5_rounded=d4/1000
// x5 or fifth approximation
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"")
```

```
    if x5_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    elseif x5_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    end
    e1=x5_rounded
    e2=horner(y,[e1])
    e3=e2*1000
    e4=ceil(e3)
    x6_rounded=e4/1000
// x6 or sixth approximation
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"")
    if x6_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    elseif x6_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    end
    f1=x6_rounded
    f2=horner(y,[f1])
    f3=f2*1000
    f4=ceil(f3)
    x7_rounded=f4/1000
// x7 or seventh approximation
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"")
    if x7_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    elseif x7_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    end
    g1=x7_rounded
    g2=horner(y,[g1])
    g3=g2*1000
    g4=ceil(g3)
    x8_rounded=g4/1000
// x8 or eighth approximation
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"")
    if x8_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
    elseif x8_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
```

```
    end
    h1=x8_rounded
    h2=horner(y,[h1])
    h3=h2*1000
    h4=ceil(h3)
    x9_rounded=h4/1000
// x9 or ninth approximation
    disp("x9 or ninth approximation:  "+string(x9_rounded)+"")
    if x9_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    elseif x9_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    end
    i1=x9_rounded
    i2=horner(y,[i1])
    i3=i2*1000
    i4=ceil(i3)
    x10_rounded=i4/1000
// x10 or tenth approximation
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"")
    if x10_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    elseif x10_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    end
    j1=x10_rounded
    j2=horner(y,[j1])
    j3=j2*1000
    j4=ceil(j3)
    x11_rounded=j4/1000
// x11 or eleventh approximation
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"")
    if x11_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    elseif x11_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    end
    k1=x11_rounded
    k2=horner(y,[k1])
    k3=k2*1000
    k4=ceil(k3)
    x12_rounded=k4/1000
// x12 or twelfth approximation
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"")
```

```
    if x12_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    elseif x12_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    end
    l1=x12_rounded
    l2=horner(y,[l1])
    l3=l2*1000
    l4=ceil(l3)
    x13_rounded=l4/1000
// x13 or thirteenth approximation
    disp("x13 or tirtheenth approximation:  "+string(x13_rounded)+"")
    if x13_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    elseif x13_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    end
    m1=x13_rounded
    m2=horner(y,[m1])
    m3=m2*1000
    m4=ceil(m3)
    x14_rounded=m4/1000
// x14 or fourteenth approximation
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"")
    if x14_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    elseif x14_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    end
    n1=x14_rounded
    n2=horner(y,[n1])
    n3=n2*1000
    n4=ceil(n3)
    x15_rounded=n4/1000
// x15 or fifteenth approximation
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"")
    if x15_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    elseif x15_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
```

```
    end
    o1=x15_rounded
    o2=horner(y,[o1])
    o3=o2*1000
    o4=ceil(o3)
    x16_rounded=o4/1000
// x16 or sixteenth approximation
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"")
    if x16_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
        abort
    elseif x16_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
        abort
    end
    p1=x16_rounded
    p2=horner(y,[p1])
    p3=p2*1000
    p4=ceil(p3)
    x17_rounded=p4/1000
// x17 or seventeenth approximation
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"")
    if x17_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    elseif x17_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    end
    q1=x17_rounded
    q2=horner(y,[q1])
    q3=q2*1000
    q4=ceil(q3)
    x18_rounded=q4/1000
// x18 or eighteenth approximation
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"")
    if x18_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    elseif x18_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    end
    r1=x18_rounded
    r2=horner(y,[r1])
    r3=r2*1000
    r4=ceil(r3)
    x19_rounded=r4/1000
// x19 or nineteenth approximation
    disp("x19 or nineteenth approximation:  "+string(x19_rounded)+"")
```

```
    if x19_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
        abort
    elseif x19_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
        abort
    end
    s1=x19_rounded
    s2=horner(y,[s1])
    s3=s2*1000
    s4=ceil(s3)
    x20_rounded=s4/1000
// x20 or twentieth approximation
    disp("x20 or twentieth approximation:  "+string(x20_rounded)+"")
    if x20_rounded==x19_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
        abort
    elseif x20_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
        abort
    end
    t1=x20_rounded
    t2=horner(y,[t1])
    t3=t2*1000
    t4=ceil(t3)
    x21_rounded=t4/1000
// x21 or twenty-first approximation
    disp("x21 or twenty-first approximation:  "+string(x21_rounded)+"")
    if x21_rounded==x20_rounded then
        disp("")
        disp("The root is "+string(x21_rounded)+"")
        abort
    elseif x21_rounded==x19_rounded then
        disp("")
        disp("The root is "+string(x21_rounded)+"")
        abort
    end
    u1=x21_rounded
    u2=horner(y,[u1])
    u3=u2*1000
    u4=ceil(u3)
    x22_rounded=u4/1000
// x22 or twenty-second approximation
    disp("x22 or twenty-second approximation:  "+string(x22_rounded)+"")
    if x22_rounded==x21_rounded then
        disp("")
        disp("The root is "+string(x22_rounded)+"")
        abort
    elseif x22_rounded==x20_rounded then
        disp("")
        disp("The root is "+string(x22_rounded)+"")
        abort
```

```
        end
        v1=x22_rounded
        v2=horner(y,[v1])
        v3=v2*1000
        v4=ceil(v3)
        x23_rounded=v4/1000
    // x23 or twenty-third approximation
        disp("x23 or twenty-third approximation:  "+string(x23_rounded)+"")
        if x23_rounded==x22_rounded then
            disp("")
            disp("The root is "+string(x23_rounded)+"")
            abort
        elseif x23_rounded==x21_rounded then
            disp("")
            disp("The root is "+string(x23_rounded)+"")
            abort
        end
        w1=x23_rounded
        w2=horner(y,[w1])
        w3=w2*1000
        w4=ceil(w3)
        x24_rounded=w4/1000
    // x24 or twenty-fourth approximation
        disp("x24 or twenty-fourth approximation:  "+string(x24_rounded)+"")
        if x24_rounded==x23_rounded then
            disp("")
            disp("The root is "+string(x24_rounded)+"")
            abort
        elseif x24_rounded==x22_rounded then
            disp("")
            disp("The root is "+string(x24_rounded)+"")
            abort
        end
        x_x1=x24_rounded
        x_x1_2=horner(y,[x_x1])
        x_x1_3=x_x1_2*1000
        x_x1_4=ceil(x_x1_3)
        x25_rounded=x_x1_4/1000
    // x25 or twenty-fifth approximation
        disp("x25 or twenty-fifth approximation:  "+string(x25_rounded)+"")
        if x25_rounded==x24_rounded then
            disp("")
            disp("The root is "+string(x25_rounded)+"")
            abort
        elseif x25_rounded==x23_rounded then
            disp("")
            disp("The root is "+string(x25_rounded)+"")
            abort
        else
            disp("")
            disp("ERROR: Cannot approximate anymore! Exceeded the maximum capabilities of the
program.")
            disp("Please wait for the next patch update of the current version!")
            disp("")
            disp("TIP: You can try to re-format your function and try inputting it using either of the two other
options avaialble under Fixed-Point Iteration Method.")
```

```
        abort
    end
// RADICAL ALGORITHM BRANCH
elseif ANSWER2==312 then
    a2=horner(y,[a1])
    a3=nthroot (a2,z_nthroot)
    a4=a3*1000
    a5=ceil(a4)
    x2_rounded=a5/1000
// x2 or second approximation
    disp("x2 or second approximation:  "+string(x2_rounded)+"")
    if x2_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x2_rounded)+"")
        abort
    end
    b1=x2_rounded
    b2=horner(y,[b1])
    b3=nthroot (b2,z_nthroot)
    b4=b3*1000
    b5=ceil(b4)
    x3_rounded=b5/1000
// x3 or third approximation
    disp("x3 or third approximation:  "+string(x3_rounded)+"")
    if x3_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    elseif x3_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    end
    c1=x3_rounded
    c2=horner(y,[c1])
    c3=nthroot (c2,z_nthroot)
    c4=c3*1000
    c5=ceil(c4)
    x4_rounded=c5/1000
// x4 or fourth approximation
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"")
    if x4_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    elseif x4_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    end
    d1=x4_rounded
    d2=horner(y,[d1])
    d3=nthroot (d2,z_nthroot)
    d4=d3*1000
    d5=ceil(d4)
    x5_rounded=d5/1000
```

```
// x5 or fifth approximation
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"")
    if x5_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    elseif x5_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    end
    e1=x5_rounded
    e2=horner(y,[e1])
    e3=nthroot (e2,z_nthroot)
    e4=e3*1000
    e5=ceil(e4)
    x6_rounded=e5/1000
// x6 or sixth approximation
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"")
    if x6_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    elseif x6_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    end
    f1=x6_rounded
    f2=horner(y,[f1])
    f3=nthroot (f2,z_nthroot)
    f4=f3*1000
    f5=ceil(f4)
    x7_rounded=f5/1000
// x7 or seventh approximation
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"")
    if x7_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    elseif x7_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    end
    g1=x7_rounded
    g2=horner(y,[g1])
    g3=nthroot (g2,z_nthroot)
    g4=g3*1000
    g5=ceil(g4)
    x8_rounded=g5/1000
// x8 or eighth approximation
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"")
    if x8_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
```

```
        abort
    elseif x8_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
    end
    h1=x8_rounded
    h2=horner(y,[h1])
    h3=nthroot (h2,z_nthroot)
    h4=h3*1000
    h5=ceil(h4)
    x9_rounded=h5/1000
// x9 or ninth approximation
    disp("x9 or ninth approximation:  "+string(x9_rounded)+"")
    if x9_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    elseif x9_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    end
    i1=x9_rounded
    i2=horner(y,[i1])
    i3=nthroot (i2,z_nthroot)
    i4=i3*1000
    i5=ceil(i4)
    x10_rounded=i5/1000
// x10 or tenth approximation
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"")
    if x10_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    elseif x10_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    end
    j1=x10_rounded
    j2=horner(y,[j1])
    j3=nthroot (j2,z_nthroot)
    j4=j3*1000
    j5=ceil(j4)
    x11_rounded=j5/1000
// x11 or eleventh approximation
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"")
    if x11_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    elseif x11_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
```

```
    end
    k1=x11_rounded
    k2=horner(y,[k1])
    k3=nthroot (k2,z_nthroot)
    k4=k3*1000
    k5=ceil(k4)
    x12_rounded=k5/1000
// x12 or twelfth approximation
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"")
    if x12_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    elseif x12_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    end
    l1=x12_rounded
    l2=horner(y,[l1])
    l3=nthroot (l2,z_nthroot)
    l4=l3*1000
    l5=ceil(l4)
    x13_rounded=l5/1000
// x13 or thirteenth approximation
    disp("x13 or tirtheenth approximation:  "+string(x13_rounded)+"")
    if x13_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    elseif x13_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    end
    m1=x13_rounded
    m2=horner(y,[m1])
    m3=nthroot (m2,z_nthroot)
    m4=m3*1000
    m5=ceil(m4)
    x14_rounded=m5/1000
// x14 or fourteenth approximation
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"")
    if x14_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    elseif x14_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    end
    n1=x14_rounded
    n2=horner(y,[n1])
    n3=nthroot (n2,z_nthroot)
    n4=n3*1000
```

```
    n5=ceil(n4)
    x15_rounded=n5/1000
// x15 or fifteenth approximation
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"")
    if x15_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    elseif x15_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    end
    o1=x15_rounded
    o2=horner(y,[o1])
    o3=nthroot (o2,z_nthroot)
    o4=o3*1000
    o5=ceil(o4)
    x16_rounded=o5/1000
// x16 or sixteenth approximation
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"")
    if x16_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
        abort
    elseif x16_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
        abort
    end
    p1=x16_rounded
    p2=horner(y,[p1])
    p3=nthroot (p2,z_nthroot)
    p4=p3*1000
    p5=ceil(p4)
    x17_rounded=p5/1000
// x17 or seventeenth approximation
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"")
    if x17_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    elseif x17_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    end
    q1=x17_rounded
    q2=horner(y,[q1])
    q3=nthroot (q2,z_nthroot)
    q4=q3*1000
    q5=ceil(q4)
    x18_rounded=q5/1000
// x18 or eighteenth approximation
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"")
    if x18_rounded==x17_rounded then
```

```
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    elseif x18_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    end
    r1=x18_rounded
    r2=horner(y,[r1])
    r3=nthroot (r2,z_nthroot)
    r4=r3*1000
    r5=ceil(r4)
    x19_rounded=r5/1000
// x19 or nineteenth approximation
    disp("x19 or nineteenth approximation:  "+string(x19_rounded)+"")
    if x19_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
        abort
    elseif x19_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
        abort
    end
    s1=x19_rounded
    s2=horner(y,[s1])
    s3=nthroot (s2,z_nthroot)
    s4=s3*1000
    s5=ceil(s4)
    x20_rounded=s5/1000
// x20 or twentieth approximation
    disp("x20 or twentieth approximation:  "+string(x20_rounded)+"")
    if x20_rounded==x19_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
        abort
    elseif x20_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
        abort
    end
    t1=x20_rounded
    t2=horner(y,[t1])
    t3=nthroot (t2,z_nthroot)
    t4=t3*1000
    t5=ceil(t4)
    x21_rounded=t5/1000
// x21 or twenty-first approximation
    disp("x21 or twenty-first approximation:  "+string(x21_rounded)+"")
    if x21_rounded==x20_rounded then
        disp("")
        disp("The root is "+string(x21_rounded)+"")
        abort
    elseif x21_rounded==x19_rounded then
        disp("")
```

```
        disp("The root is "+string(x21_rounded)+"")
        abort
    end
    u1=x21_rounded
    u2=horner(y,[u1])
    u3=nthroot (u2,z_nthroot)
    u4=u3*1000
    u5=ceil(u4)
    x22_rounded=u5/1000
// x22 or twenty-second approximation
    disp("x22 or twenty-second approximation:  "+string(x22_rounded)+"")
    if x22_rounded==x21_rounded then
        disp("")
        disp("The root is "+string(x22_rounded)+"")
        abort
    elseif x22_rounded==x20_rounded then
        disp("")
        disp("The root is "+string(x22_rounded)+"")
        abort
    end
    v1=x22_rounded
    v2=horner(y,[v1])
    v3=nthroot (v2,z_nthroot)
    v4=v3*1000
    v5=ceil(v4)
    x23_rounded=v5/1000
// x23 or twenty-third approximation
    disp("x23 or twenty-third approximation:  "+string(x23_rounded)+"")
    if x23_rounded==x22_rounded then
        disp("")
        disp("The root is "+string(x23_rounded)+"")
        abort
    elseif x23_rounded==x21_rounded then
        disp("")
        disp("The root is "+string(x23_rounded)+"")
        abort
    end
    w1=x23_rounded
    w2=horner(y,[w1])
    w3=nthroot (w2,z_nthroot)
    w4=w3*1000
    w5=ceil(w4)
    x24_rounded=w5/1000
// x24 or twenty-fourth approximation
    disp("x24 or twenty-fourth approximation:  "+string(x24_rounded)+"")
    if x24_rounded==x23_rounded then
        disp("")
        disp("The root is "+string(x24_rounded)+"")
        abort
    elseif x24_rounded==x22_rounded then
        disp("")
        disp("The root is "+string(x24_rounded)+"")
        abort
    end
    x_x1=x24_rounded
    x_x1_2=horner(y,[x_x1])
```

```
            x_x1_3=nthroot (x_x1_2,z_nthroot)
            x_x1_4=x_x1_3*1000
            x_x1_5=ceil(x_x1_4)
            x25_rounded=x_x1_5/1000
        // x25 or twenty-fifth approximation
            disp("x25 or twenty-fifth approximation:  "+string(x25_rounded)+"")
            if x25_rounded==x24_rounded then
                disp("")
                disp("The root is "+string(x25_rounded)+"")
                abort
            elseif x25_rounded==x23_rounded then
                disp("")
                disp("The root is "+string(x25_rounded)+"")
                abort
            else
                disp("")
                disp("ERROR: Cannot approximate anymore! Exceeded the maximum capabilities of the
program.")
                disp("Please wait for the next patch update of the current version!")
                disp("")
                disp("TIP: You can try to re-format your function and try inputting it using either of the two other
options avaialble under Fixed-Point Iteration Method.")
                abort
            end
        else
            clc
            disp("ERROR: Did not follow given instructions.")
            disp("Please restart the program and try again.")
            abort
        end
    elseif ANSWER1==321 then
        clc
        disp("Mode: Newton-Raphson Method")
        disp("How to Use G6-IOMRA 1.0.1: Input polynomial function in this format: poly([a,b,c,--
>nth],%x%,%coeff%)")
        disp("Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desired
polynomial function in an ascending order in terms of degree.")
        disp("")
        disp("(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab
coding mechanics so please replace % with quotation mark!)")
        disp("")
        disp("e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2")
        disp("")
        y=input (" Input your f(x):  ")
        y_prime=input (" Input derivative of f(x):  ")
        x=input (" Input your x0:  ") // x=input = x(0) or initial approximation
        disp("x0 or initial approximation:  "+string(x)+"")
        y1=horner(y,[x])
        y2=y1*1000
        y3=int(y2)
        y_rounded=y3/1000 // y_rounded = f(x0)
        disp("f(x0):  "+string(y_rounded)+"")
        y1_prime=horner(y_prime,[x])
        y2_prime=y1_prime*1000
        y3_prime=ceil(y2_prime)
        y_prime_rounded=y3_prime/1000 // y_prime_rounded = f'(x0)
```

```
    disp("first derivative of f(x0):  "+string(y_prime_rounded)+"")
    disp("")
// x1 or first approximation
    x1=((x)-((y_rounded)/(y_prime_rounded)))
    x1_1=x1*1000
    x1_2=ceil(x1_1)
    x1_rounded=x1_2/1000
    disp("x1 or first approximation:  "+string(x1_rounded)+"")
    a1=horner(y,[x1_rounded])
    a2=a1*1000
    a3=ceil(a2)
    a_rounded=a3/1000 // a_rounded = f(x1)
    disp("f(x1):  "+string(a_rounded)+"")
    a1_prime=horner(y_prime,[x1_rounded])
    a2_prime=a1_prime*1000
    a3_prime=ceil(a2_prime)
    a_prime_rounded=a3_prime/1000 // a_prime_rounded = f'(x1)
    disp("first derivative of f(x1):  "+string(a_prime_rounded)+"")
    disp("")
// x2 or second approximation
    x2=((x1)-((a_rounded)/(a_prime_rounded)))
    x2_1=x2*1000
    x2_2=ceil(x2_1)
    x2_rounded=x2_2/1000
    disp("x2 or second approximation:  "+string(x2_rounded)+"")
    if x2_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x2_rounded)+"")
        abort
    end
    b1=horner(y,[x2_rounded])
    b2=b1*1000
    b3=ceil(b2)
    b_rounded=b3/1000 // b_rounded = f(x2)
    disp("f(x2):  "+string(b_rounded)+"")
    b1_prime=horner(y_prime,[x2_rounded])
    b2_prime=b1_prime*1000
    b3_prime=ceil(b2_prime)
    b_prime_rounded=b3_prime/1000 // b_prime_rounded = f'(x2)
    disp("first derivative of f(x2):  "+string(b_prime_rounded)+"")
    disp("")
// x3 or third approximation
    x3=((x2)-((b_rounded)/(b_prime_rounded)))
    x3_1=x3*1000
    x3_2=ceil(x3_1)
    x3_rounded=x3_2/1000
    disp("x3 or third approximation:  "+string(x3_rounded)+"")
    if x3_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    elseif x3_rounded==x1_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    end
```

```
    c1=horner(y,[x3_rounded])
    c2=c1*1000
    c3=ceil(c2)
    c_rounded=c3/1000 // c_rounded = f(x3)
    disp("f(x3):  "+string(c_rounded)+"")
    c1_prime=horner(y_prime,[x3_rounded])
    c2_prime=c1_prime*1000
    c3_prime=ceil(c2_prime)
    c_prime_rounded=c3_prime/1000 // c_prime_rounded = f'(x3)
    disp("first derivative of f(x3):  "+string(c_prime_rounded)+"")
    disp("")
// x4 or fourth approximation
    x4=((x3)-((c_rounded)/(c_prime_rounded)))
    x4_1=x4*1000
    x4_2=ceil(x4_1)
    x4_rounded=x4_2/1000
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"")
    if x4_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    elseif x4_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    end
    d1=horner(y,[x4_rounded])
    d2=d1*1000
    d3=ceil(d2)
    d_rounded=d3/1000 // d_rounded = f(x4)
    disp("f(x4):  "+string(d_rounded)+"")
    d1_prime=horner(y_prime,[x4_rounded])
    d2_prime=d1_prime*1000
    d3_prime=ceil(d2_prime)
    d_prime_rounded=d3_prime/1000 // d_prime_rounded = f'(x4)
    disp("first derivative of f(x4):  "+string(d_prime_rounded)+"")
    disp("")
// x5 or fifth approximation
    x5=((x4)-((d_rounded)/(d_prime_rounded)))
    x5_1=x5*1000
    x5_2=ceil(x5_1)
    x5_rounded=x5_2/1000
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"")
    if x5_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    elseif x5_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
        abort
    end
    e1=horner(y,[x5_rounded])
    e2=e1*1000
    e3=ceil(e2)
    e_rounded=e3/1000 // e_rounded = f(x5)
```

```
    disp("f(x5):  "+string(e_rounded)+"")
    e1_prime=horner(y_prime,[x5_rounded])
    e2_prime=e1_prime*1000
    e3_prime=ceil(e2_prime)
    e_prime_rounded=e3_prime/1000 // e_prime_rounded = f'(x5)
    disp("first derivative of f(x5):  "+string(e_prime_rounded)+"")
    disp("")
// x6 or sixth approximation
    x6=((x5)-((e_rounded)/(e_prime_rounded)))
    x6_1=x6*1000
    x6_2=ceil(x6_1)
    x6_rounded=x6_2/1000
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"")
    if x6_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    elseif x6_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    end
    f1=horner(y,[x6_rounded])
    f2=f1*1000
    f3=ceil(f2)
    f_rounded=f3/1000 // f_rounded = f(x6)
    disp("f(x6):  "+string(f_rounded)+"")
    f1_prime=horner(y_prime,[x6_rounded])
    f2_prime=f1_prime*1000
    f3_prime=ceil(f2_prime)
    f_prime_rounded=f3_prime/1000 // f_prime_rounded = f'(x6)
    disp("first derivative of f(x6):  "+string(f_prime_rounded)+"")
    disp("")
// x7 or seventh approximation
    x7=((x6)-((f_rounded)/(f_prime_rounded)))
    x7_1=x7*1000
    x7_2=ceil(x7_1)
    x7_rounded=x7_2/1000
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"")
    if x7_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    elseif x7_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    end
    g1=horner(y,[x7_rounded])
    g2=g1*1000
    g3=ceil(g2)
    g_rounded=g3/1000 // g_rounded = f(x7)
    disp("f(x7):  "+string(g_rounded)+"")
    g1_prime=horner(y_prime,[x7_rounded])
    g2_prime=g1_prime*1000
    g3_prime=ceil(g2_prime)
```

```
    g_prime_rounded=g3_prime/1000 // g_prime_rounded = f'(x7)
    disp("first derivative of f(x7):  "+string(g_prime_rounded)+"")
    disp("")
// x8 or eighth approximation
    x8=((x7)-((g_rounded)/(g_prime_rounded)))
    x8_1=x8*1000
    x8_2=ceil(x8_1)
    x8_rounded=x8_2/1000
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"")
    if x8_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
    elseif x8_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
    end
    h1=horner(y,[x8_rounded])
    h2=h1*1000
    h3=ceil(h2)
    h_rounded=h3/1000 // h_rounded = f(x8)
    disp("f(x8):  "+string(h_rounded)+"")
    h1_prime=horner(y_prime,[x8_rounded])
    h2_prime=h1_prime*1000
    h3_prime=ceil(h2_prime)
    h_prime_rounded=h3_prime/1000 // h_prime_rounded = f'(x8)
    disp("first derivative of f(x8):  "+string(h_prime_rounded)+"")
    disp("")
// x9 or nineteenth approximation
    x9=((x8)-((h_rounded)/(h_prime_rounded)))
    x9_1=x9*1000
    x9_2=ceil(x9_1)
    x9_rounded=x9_2/1000
    disp("x9 or nineteenth approximation:  "+string(x9_rounded)+"")
    if x9_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    elseif x9_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    end
    i1=horner(y,[x9_rounded])
    i2=i1*1000
    i3=ceil(i2)
    i_rounded=i3/1000 // i_rounded = f(x9)
    disp("f(x9):  "+string(i_rounded)+"")
    i1_prime=horner(y_prime,[x9_rounded])
    i2_prime=i1_prime*1000
    i3_prime=ceil(i2_prime)
    i_prime_rounded=i3_prime/1000 // i_prime_rounded = f'(x9)
    disp("first derivative of f(x9):  "+string(i_prime_rounded)+"")
    disp("")
// x10 or tenth approximation
```

```
    x10=((x9)-((i_rounded)/(i_prime_rounded)))
    x10_1=x10*1000
    x10_2=ceil(x10_1)
    x10_rounded=x10_2/1000
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"")
    if x10_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    elseif x10_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    end
    j1=horner(y,[x10_rounded])
    j2=j1*1000
    j3=ceil(j2)
    j_rounded=j3/1000 // j_rounded = f(x10)
    disp("f(x10):  "+string(j_rounded)+"")
    j1_prime=horner(y_prime,[x10_rounded])
    j2_prime=j1_prime*1000
    j3_prime=ceil(j2_prime)
    j_prime_rounded=j3_prime/1000 // j_prime_rounded = f'(x10)
    disp("first derivative of f(x10):  "+string(j_prime_rounded)+"")
    disp("")
// x11 or eleventh approximation
    x11=((x10)-((j_rounded)/(j_prime_rounded)))
    x11_1=x11*1000
    x11_2=ceil(x11_1)
    x11_rounded=x11_2/1000
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"")
    if x11_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    elseif x11_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    end
    k1=horner(y,[x11_rounded])
    k2=k1*1000
    k3=ceil(k2)
    k_rounded=k3/1000 // k_rounded = f(x11)
    disp("f(x11):  "+string(k_rounded)+"")
    k1_prime=horner(y_prime,[x11_rounded])
    k2_prime=k1_prime*1000
    k3_prime=ceil(k2_prime)
    k_prime_rounded=k3_prime/1000 // k_prime_rounded = f'(x11)
    disp("first derivative of f(x11):  "+string(k_prime_rounded)+"")
    disp("")
// x12 or twelfth approximation
    x12=((x11)-((k_rounded)/(k_prime_rounded)))
    x12_1=x12*1000
    x12_2=ceil(x12_1)
    x12_rounded=x12_2/1000
```

```
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"")
    if x12_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    elseif x12_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x12_rounded)+"")
        abort
    end
    l1=horner(y,[x12_rounded])
    l2=l1*1000
    l3=ceil(l2)
    l_rounded=l3/1000 // l_rounded = f(x12)
    disp("f(x12):  "+string(l_rounded)+"")
    l1_prime=horner(y_prime,[x12_rounded])
    l2_prime=l1_prime*1000
    l3_prime=ceil(l2_prime)
    l_prime_rounded=l3_prime/1000 // l_prime_rounded = f'(x12)
    disp("first derivative of f(x12):  "+string(l_prime_rounded)+"")
    disp("")
// x13 or thirteenth approximation
    x13=((x12)-((l_rounded)/(l_prime_rounded)))
    x13_1=x13*1000
    x13_2=ceil(x13_1)
    x13_rounded=x13_2/1000
    disp("x13 or thirteenth approximation:  "+string(x13_rounded)+"")
    if x13_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    elseif x13_rounded==x11_rounded then
        disp("")
        disp("The root is "+string(x13_rounded)+"")
        abort
    end
    m1=horner(y,[x13_rounded])
    m2=m1*1000
    m3=ceil(m2)
    m_rounded=m3/1000 // m_rounded = f(x13)
    disp("f(x13):  "+string(m_rounded)+"")
    m1_prime=horner(y_prime,[x13_rounded])
    m2_prime=m1_prime*1000
    m3_prime=ceil(m2_prime)
    m_prime_rounded=m3_prime/1000 // m_prime_rounded = f'(x13)
    disp("first derivative of f(x13):  "+string(m_prime_rounded)+"")
    disp("")
// x14 or fourteenth approximation
    x14=((x13)-((m_rounded)/(m_prime_rounded)))
    x14_1=x14*1000
    x14_2=ceil(x14_1)
    x14_rounded=x14_2/1000
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"")
    if x14_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
```

```
        abort
    elseif x14_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    end
    n1=horner(y,[x14_rounded])
    n2=n1*1000
    n3=ceil(n2)
    n_rounded=n3/1000 // n_rounded = f(x14)
    disp("f(x14):  "+string(n_rounded)+"")
    n1_prime=horner(y_prime,[x14_rounded])
    n2_prime=n1_prime*1000
    n3_prime=ceil(n2_prime)
    n_prime_rounded=n3_prime/1000 // n_prime_rounded = f'(x14)
    disp("first derivative of f(x14):  "+string(n_prime_rounded)+"")
    disp("")
// x15 or fifteenth approximation
    x15=((x14)-((n_rounded)/(n_prime_rounded)))
    x15_1=x15*1000
    x15_2=ceil(x15_1)
    x15_rounded=x15_2/1000
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"")
    if x15_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    elseif x15_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    end
    o1=horner(y,[x15_rounded])
    o2=o1*1000
    o3=ceil(o2)
    o_rounded=o3/1000 // o_rounded = f(x15)
    disp("f(x15):  "+string(o_rounded)+"")
    o1_prime=horner(y_prime,[x15_rounded])
    o2_prime=o1_prime*1000
    o3_prime=ceil(o2_prime)
    o_prime_rounded=o3_prime/1000 // o_prime_rounded = f'(x15)
    disp("first derivative of f(x15):  "+string(o_prime_rounded)+"")
    disp("")
// x16 or sixteenth approximation
    x16=((x15)-((o_rounded)/(o_prime_rounded)))
    x16_1=x16*1000
    x16_2=ceil(x16_1)
    x16_rounded=x16_2/1000
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"")
    if x16_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
        abort
    elseif x16_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
```

```
        abort
    end
    p1=horner(y,[x16_rounded])
    p2=p1*1000
    p3=ceil(p2)
    p_rounded=p3/1000 // p_rounded = f(x16)
    disp("f(x16):  "+string(p_rounded)+"")
    p1_prime=horner(y_prime,[x16_rounded])
    p2_prime=p1_prime*1000
    p3_prime=ceil(p2_prime)
    p_prime_rounded=p3_prime/1000 // p_prime_rounded = f'(x16)
    disp("first derivative of f(x16):  "+string(p_prime_rounded)+"")
    disp("")
// x17 or seventeenth approximation
    x17=((x16)-((p_rounded)/(p_prime_rounded)))
    x17_1=x17*1000
    x17_2=ceil(x17_1)
    x17_rounded=x17_2/1000
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"")
    if x17_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    elseif x17_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    end
    q1=horner(y,[x17_rounded])
    q2=q1*1000
    q3=ceil(q2)
    q_rounded=q3/1000 // q_rounded = f(x17)
    disp("f(x17):  "+string(q_rounded)+"")
    q1_prime=horner(y_prime,[x17_rounded])
    q2_prime=q1_prime*1000
    q3_prime=ceil(q2_prime)
    q_prime_rounded=q3_prime/1000 // q_prime_rounded = f'(x17)
    disp("first derivative of f(x17):  "+string(q_prime_rounded)+"")
    disp("")
// x18 or eighteenth approximation
    x18=((x17)-((q_rounded)/(q_prime_rounded)))
    x18_1=x18*1000
    x18_2=ceil(x18_1)
    x18_rounded=x18_2/1000
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"")
    if x18_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    elseif x18_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    end
    r1=horner(y,[x18_rounded])
    r2=r1*1000
```

```
        r3=ceil(r2)
        r_rounded=r3/1000 // r_rounded = f(x18)
        disp("f(x18):  "+string(r_rounded)+"")
        r1_prime=horner(y_prime,[x18_rounded])
        r2_prime=r1_prime*1000
        r3_prime=ceil(r2_prime)
        r_prime_rounded=r3_prime/1000 // r_prime_rounded = f'(x18)
        disp("first derivative of f(x18):  "+string(r_prime_rounded)+"")
        disp("")
    // x19 or nineteenth approximation
        x19=((x18)-((r_rounded)/(r_prime_rounded)))
        x19_1=x19*1000
        x19_2=ceil(x19_1)
        x19_rounded=x19_2/1000
        disp("x19 or nineteenth approximation:  "+string(x19_rounded)+"")
        if x19_rounded==x18_rounded then
            disp("")
            disp("The root is "+string(x19_rounded)+"")
            abort
        elseif x19_rounded==x17_rounded then
            disp("")
            disp("The root is "+string(x19_rounded)+"")
            abort
        end
        s1=horner(y,[x19_rounded])
        s2=s1*1000
        s3=ceil(s2)
        s_rounded=s3/1000 // s_rounded = f(x19)
        disp("f(x19):  "+string(s_rounded)+"")
        s1_prime=horner(y_prime,[x18_rounded])
        s2_prime=s1_prime*1000
        s3_prime=ceil(s2_prime)
        s_prime_rounded=s3_prime/1000 // s_prime_rounded = f'(x19)
        disp("first derivative of f(x19):  "+string(s_prime_rounded)+"")
        disp("")
    // x20 or twentieth approximation
        x20=((x19)-((s_rounded)/(s_prime_rounded)))
        x20_1=x20*1000
        x20_2=ceil(x20_1)
        x20_rounded=x20_2/1000
        disp("x20 or twentieth approximation:  "+string(x20_rounded)+"")
        if x20_rounded==x19_rounded then
            disp("")
            disp("The root is "+string(x20_rounded)+"")
            abort
        elseif x20_rounded==x18_rounded then
            disp("")
            disp("The root is "+string(x20_rounded)+"")
            abort
        else
            disp("")
            disp("ERROR: Cannot approximate anymore! Exceeded the maximum capabilities of the
program.")
            disp("Please wait for the next patch update of the current version!")
            abort
        end
```

```
    elseif ANSWER1==123 then
        clc
        disp("Mode: Secant Method")
        disp("How to Use G6-IOMRA 1.0.1: Input polynomial function in this format: poly([a,b,c,-->nth],%x%,%coeff%)")
        disp("Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desired polynomial function in an ascending order in terms of degree.")
        disp("")
        disp("(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab coding mechanics so please replace % with quotation mark!)")
        disp("")
        disp("e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2")
        disp("")
        y=input (" Input your f(x):  ")
        x=input (" Input your x0:  ") // x=input = x(0) or initial approximation
        x1=input(" Input your x1:  ") // x1=input = x(1) or first approximation
    // First Iteration
        x_1=horner(y,[x])
        x_2=x_1*1000
        x_3=ceil(x_2)
        y_rounded=x_3/1000
        x1_1=horner(y,[x1])
        x1_2=x1_1*1000
        x1_3=ceil(x1_2)
        y1_rounded=x1_3/1000
        disp("")
        disp("Iteration 1:")
        disp("x0 or initial approximation:  "+string(x)+"") // Xn-1
        disp("x1 or first approximation:  "+string(x1)+"") // Xn
        disp("f(x0):  "+string(y_rounded)+"") // f(Xn-1)
        disp("f(x1):  "+string(y1_rounded)+"") // f(Xn)
        disp("")
    // Second Iteration
        x2=((x1)-(((y1_rounded)(x1-x))/((y1_rounded)-(y_rounded))))
        x2_1=x2*1000
        x2_2=ceil(x2_1)
        x2_rounded=x2_2/1000
        disp("x2 or second approximation:  "+string(x2_rounded)+"")
        if x2_rounded==x1 then
            disp("")
            disp("The root is "+string(x2_rounded)+"")
            abort
        end
        y2=horner(y,[x2_rounded])
        y2_1=y2*1000
        y2_2=ceil(y2_1)
        y2_rounded=y2_2/1000
        disp("")
        disp("Iteration 2:")
        disp("x1 or first approximation:  "+string(x1)+"") // Xn-1
        disp("x2 or second approximation:  "+string(x2_rounded)+"") // Xn
        disp("f(x1):  "+string(y1_rounded)+"") // f(Xn-1)
        disp("f(x2):  "+string(y2_rounded)+"") // f(Xn)
        disp("")
    // Third Iteration
        x3=((x2_rounded)-(((y2_rounded)*(x2_rounded-x1))/((y2_rounded)-(y1_rounded))))
```

```
    x3_1=x3*1000
    x3_2=ceil(x3_1)
    x3_rounded=x3_2/1000
    disp("x3 or third approximation:  "+string(x3_rounded)+"")
    if x3_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
        abort
    elseif x3_rounded==x1 then
        disp("")
        disp("The root is "+string(x3_rounded)+"")
    end
    y3=horner(y,[x3_rounded])
    y3_1=y3*1000
    y3_2=ceil(y3_1)
    y3_rounded=y3_2/1000
    disp("")
    disp("Iteration 3:")
    disp("x2 or second approximation:  "+string(x2_rounded)+"") // Xn-1
    disp("x3 or third approximation:  "+string(x3_rounded)+"") // Xn
    disp("f(x2):  "+string(y2_rounded)+"") // f(Xn-1)
    disp("f(x3):  "+string(y3_rounded)+"") // f(Xn)
    disp("")
// Fourth Iteration
    x4=((x3_rounded)-(((y3_rounded)*(x3_rounded-x2_rounded))/((y3_rounded)-(y2_rounded))))
    x4_1=x4*1000
    x4_2=ceil(x4_1)
    x4_rounded=x4_2/1000
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"")
    if x4_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
        abort
    elseif x4_rounded==x2_rounded then
        disp("")
        disp("The root is "+string(x4_rounded)+"")
    end
    y4=horner(y,[x4_rounded])
    y4_1=y4*1000
    y4_2=ceil(y4_1)
    y4_rounded=y4_2/1000
    disp("")
    disp("Iteration 4:")
    disp("x3 or third approximation:  "+string(x3_rounded)+"") // Xn-1
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"") // Xn
    disp("f(x3):  "+string(y3_rounded)+"") // f(Xn-1)
    disp("f(x4):  "+string(y4_rounded)+"") // f(Xn)
    disp("")
// Fifth Iteration
    x5=((x4_rounded)-(((y4_rounded)*(x4_rounded-x3_rounded))/((y4_rounded)-(y3_rounded))))
    x5_1=x5*1000
    x5_2=ceil(x5_1)
    x5_rounded=x5_2/1000
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"")
    if x5_rounded==x4_rounded then
        disp("")
```

```
        disp("The root is "+string(x5_rounded)+"")
        abort
    elseif x5_rounded==x3_rounded then
        disp("")
        disp("The root is "+string(x5_rounded)+"")
    end
    y5=horner(y,[x5_rounded])
    y5_1=y5*1000
    y5_2=ceil(y5_1)
    y5_rounded=y5_2/1000
    disp("")
    disp("Iteration 5:")
    disp("x4 or fourth approximation:  "+string(x4_rounded)+"") // Xn-1
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"") // Xn
    disp("f(x4):  "+string(y4_rounded)+"") // f(Xn-1)
    disp("f(x5):  "+string(y5_rounded)+"") // f(Xn)
    disp("")
// Sixth Iteration
    x6=((x5_rounded)-(((y5_rounded)*(x5_rounded-x4_rounded))/((y5_rounded)-(y4_rounded))))
    x6_1=x6*1000
    x6_2=ceil(x6_1)
    x6_rounded=x6_2/1000
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"")
    if x6_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
        abort
    elseif x5_rounded==x4_rounded then
        disp("")
        disp("The root is "+string(x6_rounded)+"")
    end
    y6=horner(y,[x6_rounded])
    y6_1=y6*1000
    y6_2=ceil(y6_1)
    y6_rounded=y6_2/1000
    disp("")
    disp("Iteration 6:")
    disp("x5 or fifth approximation:  "+string(x5_rounded)+"") // Xn-1
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"") // Xn
    disp("f(x5):  "+string(y5_rounded)+"") // f(Xn-1)
    disp("f(x6):  "+string(y6_rounded)+"") // f(Xn)
    disp("")
// Seventh Iteration
    x7=((x6_rounded)-(((y6_rounded)*(x6_rounded-x5_rounded))/((y6_rounded)-(y5_rounded))))
    x7_1=x7*1000
    x7_2=ceil(x7_1)
    x7_rounded=x7_2/1000
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"")
    if x7_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
        abort
    elseif x7_rounded==x5_rounded then
        disp("")
        disp("The root is "+string(x7_rounded)+"")
    end
```

```
    y7=horner(y,[x7_rounded])
    y7_1=y7*1000
    y7_2=ceil(y7_1)
    y7_rounded=y7_2/1000
    disp("")
    disp("Iteration 7:")
    disp("x6 or sixth approximation:  "+string(x6_rounded)+"") // Xn-1
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"") // Xn
    disp("f(x6):  "+string(y6_rounded)+"") // f(Xn-1)
    disp("f(x7):  "+string(y7_rounded)+"") // f(Xn)
    disp("")
// Eighth Iteration
    x8=((x7_rounded)-(((y7_rounded)*(x7_rounded-x6_rounded))/((y7_rounded)-(y6_rounded))))
    x8_1=x8*1000
    x8_2=ceil(x8_1)
    x8_rounded=x8_2/1000
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"")
    if x8_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
        abort
    elseif x8_rounded==x6_rounded then
        disp("")
        disp("The root is "+string(x8_rounded)+"")
    end
    y8=horner(y,[x8_rounded])
    y8_1=y8*1000
    y8_2=ceil(y8_1)
    y8_rounded=y8_2/1000
    disp("")
    disp("Iteration 8:")
    disp("x7 or seventh approximation:  "+string(x7_rounded)+"") // Xn-1
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"") // Xn
    disp("f(x7):  "+string(y7_rounded)+"") // f(Xn-1)
    disp("f(x8):  "+string(y8_rounded)+"") // f(Xn)
    disp("")
// Ninth Iteration
    x9=((x8_rounded)-(((y8_rounded)*(x8_rounded-x7_rounded))/((y8_rounded)-(y7_rounded))))
    x9_1=x9*1000
    x9_2=ceil(x9_1)
    x9_rounded=x9_2/1000
    disp("x9 or ninth approximation:  "+string(x9_rounded)+"")
    if x9_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
        abort
    elseif x9_rounded==x7_rounded then
        disp("")
        disp("The root is "+string(x9_rounded)+"")
    end
    y9=horner(y,[x9_rounded])
    y9_1=y9*1000
    y9_2=ceil(y9_1)
    y9_rounded=y9_2/1000
    disp("")
    disp("Iteration 9:")
```

```
    disp("x8 or eighth approximation:  "+string(x8_rounded)+"") // Xn-1
    disp("x9 or ninth approximation:  "+string(x9_rounded)+"") // Xn
    disp("f(x8):  "+string(y8_rounded)+"") // f(Xn-1)
    disp("f(x9):  "+string(y9_rounded)+"") // f(Xn)
    disp("")
// Tenth Iteration
    x10=((x9_rounded)-(((y9_rounded)*(x9_rounded-x8_rounded))/((y9_rounded)-(y8_rounded))))
    x10_1=x10*1000
    x10_2=ceil(x10_1)
    x10_rounded=x10_2/1000
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"")
    if x10_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
        abort
    elseif x10_rounded==x8_rounded then
        disp("")
        disp("The root is "+string(x10_rounded)+"")
    end
    y10=horner(y,[x10_rounded])
    y10_1=y10*1000
    y10_2=ceil(y10_1)
    y10_rounded=y10_2/1000
    disp("")
    disp("Iteration 10:")
    disp("x9 or ninth approximation:  "+string(x9_rounded)+"") // Xn-1
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"") // Xn
    disp("f(x9):  "+string(y9_rounded)+"") // f(Xn-1)
    disp("f(x10):  "+string(y10_rounded)+"") // f(Xn)
    disp("")
// Eleventh Iteration
    x11=((x10_rounded)-(((y10_rounded)*(x10_rounded-x9_rounded))/((y10_rounded)-(y9_rounded))))
    x11_1=x11*1000
    x11_2=ceil(x11_1)
    x11_rounded=x11_2/1000
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"")
    if x11_rounded==x10_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
        abort
    elseif x11_rounded==x9_rounded then
        disp("")
        disp("The root is "+string(x11_rounded)+"")
    end
    y11=horner(y,[x11_rounded])
    y11_1=y11*1000
    y11_2=ceil(y11_1)
    y11_rounded=y11_2/1000
    disp("")
    disp("Iteration 11:")
    disp("x10 or tenth approximation:  "+string(x10_rounded)+"") // Xn-1
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"") // Xn
    disp("f(x10):  "+string(y10_rounded)+"") // f(Xn-1)
    disp("f(x11):  "+string(y11_rounded)+"") // f(Xn)
    disp("")
// Twelfth Iteration
```

```
    x12=((x11_rounded)-(((y11_rounded)*(x11_rounded-x10_rounded))/((y11_rounded)-
(y10_rounded))))
    x12_1=x12*1000
    x12_2=ceil(x12_1)
    x12_rounded=x12_2/1000
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"")
    if x12_rounded==x11_rounded then
       disp("")
       disp("The root is "+string(x12_rounded)+"")
       abort
    elseif x12_rounded==x10_rounded then
       disp("")
       disp("The root is "+string(x12_rounded)+"")
    end
    y12=horner(y,[x12_rounded])
    y12_1=y12*1000
    y12_2=ceil(y12_1)
    y12_rounded=y12_2/1000
    disp("")
    disp("Iteration 12:")
    disp("x11 or eleventh approximation:  "+string(x11_rounded)+"") // Xn-1
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"") // Xn
    disp("f(x11):  "+string(y11_rounded)+"") // f(Xn-1)
    disp("f(x12):  "+string(y12_rounded)+"") // f(Xn)
    disp("")
  // Thirteenth Iteration
    x13=((x12_rounded)-(((y12_rounded)*(x12_rounded-x11_rounded))/((y12_rounded)-
(y11_rounded))))
    x13_1=x13*1000
    x13_2=ceil(x13_1)
    x13_rounded=x13_2/1000
    disp("x13 or thirteenth approximation:  "+string(x13_rounded)+"")
    if x13_rounded==x12_rounded then
       disp("")
       disp("The root is "+string(x13_rounded)+"")
       abort
    elseif x13_rounded==x11_rounded then
       disp("")
       disp("The root is "+string(x13_rounded)+"")
    end
    y13=horner(y,[x13_rounded])
    y13_1=y13*1000
    y13_2=ceil(y13_1)
    y13_rounded=y13_2/1000
    disp("")
    disp("Iteration 13:")
    disp("x12 or twelfth approximation:  "+string(x12_rounded)+"") // Xn-1
    disp("x13 or thirteenth approximation:  "+string(x13_rounded)+"") // Xn
    disp("f(x12):  "+string(y12_rounded)+"") // f(Xn-1)
    disp("f(x13):  "+string(y13_rounded)+"") // f(Xn)
    disp("")
  // Fourteenth Iteration
    x14=((x13_rounded)-(((y13_rounded)*(x13_rounded-x12_rounded))/((y13_rounded)-
(y12_rounded))))
    x14_1=x14*1000
    x14_2=ceil(x14_1)
```

```
    x14_rounded=x14_2/1000
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"")
    if x14_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
        abort
    elseif x14_rounded==x12_rounded then
        disp("")
        disp("The root is "+string(x14_rounded)+"")
    end
    y14=horner(y,[x14_rounded])
    y14_1=y14*1000
    y14_2=ceil(y14_1)
    y14_rounded=y14_2/1000
    disp("")
    disp("Iteration 14:")
    disp("x13 or thirteenth approximation:  "+string(x13_rounded)+"") // Xn-1
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"") // Xn
    disp("f(x13):  "+string(y13_rounded)+"") // f(Xn-1)
    disp("f(x14):  "+string(y14_rounded)+"") // f(Xn)
    disp("")
  // Fifteenth Iteration
    x15=((x14_rounded)-(((y14_rounded)*(x14_rounded-x13_rounded))/((y14_rounded)-
(y13_rounded))))
    x15_1=x15*1000
    x15_2=ceil(x15_1)
    x15_rounded=x15_2/1000
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"")
    if x15_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
        abort
    elseif x15_rounded==x13_rounded then
        disp("")
        disp("The root is "+string(x15_rounded)+"")
    end
    y15=horner(y,[x15_rounded])
    y15_1=y15*1000
    y15_2=ceil(y15_1)
    y15_rounded=y15_2/1000
    disp("")
    disp("Iteration 15:")
    disp("x14 or fourteenth approximation:  "+string(x14_rounded)+"") // Xn-1
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"") // Xn
    disp("f(x14):  "+string(y14_rounded)+"") // f(Xn-1)
    disp("f(x15):  "+string(y15_rounded)+"") // f(Xn)
    disp("")
  // Sixteenth Iteration
    x16=((x15_rounded)-(((y15_rounded)*(x15_rounded-x14_rounded))/((y15_rounded)-
(y14_rounded))))
    x16_1=x16*1000
    x16_2=ceil(x16_1)
    x16_rounded=x16_2/1000
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"")
    if x16_rounded==x15_rounded then
        disp("")
```

```
        disp("The root is "+string(x16_rounded)+"")
        abort
    elseif x16_rounded==x14_rounded then
        disp("")
        disp("The root is "+string(x16_rounded)+"")
    end
    y16=horner(y,[x16_rounded])
    y16_1=y16*1000
    y16_2=ceil(y16_1)
    y16_rounded=y16_2/1000
    disp("")
    disp("Iteration 16:")
    disp("x15 or fifteenth approximation:  "+string(x15_rounded)+"") // Xn-1
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"") // Xn
    disp("f(x15):  "+string(y15_rounded)+"") // f(Xn-1)
    disp("f(x16):  "+string(y16_rounded)+"") // f(Xn)
    disp("")
  // Seventh Iteration
    x17=((x16_rounded)-(((y16_rounded)*(x16_rounded-x15_rounded))/((y16_rounded)-
(y15_rounded))))
    x17_1=x17*1000
    x17_2=ceil(x17_1)
    x17_rounded=x17_2/1000
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"")
    if x17_rounded==x16_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
        abort
    elseif x17_rounded==x15_rounded then
        disp("")
        disp("The root is "+string(x17_rounded)+"")
    end
    y17=horner(y,[x17_rounded])
    y17_1=y17*1000
    y17_2=ceil(y17_1)
    y17_rounded=y17_2/1000
    disp("")
    disp("Iteration 17:")
    disp("x16 or sixteenth approximation:  "+string(x16_rounded)+"") // Xn-1
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"") // Xn
    disp("f(x16):  "+string(y16_rounded)+"") // f(Xn-1)
    disp("f(x17):  "+string(y17_rounded)+"") // f(Xn)
    disp("")
  // Eighteenth Iteration
    x18=((x17_rounded)-(((y17_rounded)*(x17_rounded-x16_rounded))/((y17_rounded)-
(y16_rounded))))
    x18_1=x18*1000
    x18_2=ceil(x18_1)
    x18_rounded=x18_2/1000
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"")
    if x18_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x18_rounded)+"")
        abort
    elseif x18_rounded==x16_rounded then
        disp("")
```

```
        disp("The root is "+string(x18_rounded)+"")
    end
    y18=horner(y,[x18_rounded])
    y18_1=y18*1000
    y18_2=ceil(y18_1)
    y18_rounded=y18_2/1000
    disp("")
    disp("Iteration 18:")
    disp("x17 or seventeenth approximation:  "+string(x17_rounded)+"") // Xn-1
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"") // Xn
    disp("f(x17):  "+string(y17_rounded)+"") // f(Xn-1)
    disp("f(x18):  "+string(y18_rounded)+"") // f(Xn)
    disp("")
  // Nineteenth Iteration
    x19=((x18_rounded)-(((y18_rounded)*(x18_rounded-x17_rounded))/((y18_rounded)-
(y17_rounded))))
    x19_1=x19*1000
    x19_2=ceil(x19_1)
    x19_rounded=x19_2/1000
    disp("x19 or nineteenth approximation:  "+string(x19_rounded)+"")
    if x19_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
        abort
    elseif x19_rounded==x17_rounded then
        disp("")
        disp("The root is "+string(x19_rounded)+"")
    end
    y19=horner(y,[x19_rounded])
    y19_1=y19*1000
    y19_2=ceil(y19_1)
    y19_rounded=y19_2/1000
    disp("")
    disp("Iteration 19:")
    disp("x18 or eighteenth approximation:  "+string(x18_rounded)+"") // Xn-1
    disp("x19 or nineteenth approximation:  "+string(x19_rounded)+"") // Xn
    disp("f(x18):  "+string(y18_rounded)+"") // f(Xn-1)
    disp("f(x19):  "+string(y19_rounded)+"") // f(Xn)
    disp("")
  // Twentieth Iteration
    x20=((x19_rounded)-(((y19_rounded)*(x19_rounded-x18_rounded))/((y19_rounded)-
(y18_rounded))))
    x20_1=x20*1000
    x20_2=ceil(x20_1)
    x20_rounded=x20_2/1000
    disp("x20 or twentieth approximation:  "+string(x20_rounded)+"")
    if x20_rounded==x19_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
        abort
    elseif x20_rounded==x18_rounded then
        disp("")
        disp("The root is "+string(x20_rounded)+"")
    else
        disp("")
```

```
            disp("ERROR: Cannot approximate anymore! Exceeded the maximum capabilities of the
program.")
            disp("Please wait for the next patch update of the current version!")
            abort
        end
    else
        clc
        disp("ERROR: Did not follow given instructions.")
        disp("Please restart the program and try again.")
    end
elseif ANSWER0==0 then
    clc
    disp("Thank you for using our program!")
    disp("For any inquiries, please contact Head Developer through this email:")
    disp("bernardoraevon@gmail.com")
    abort
end
```

## III. Program Instructions

**How to Use G6-IOMRA 1.0.1:**

Input polynomial functions in this format:

poly([a,b,c,-->nth],”x”,”coeff”)

**Wherein:** Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desired polynomial function in an ascending order in terms of degree.

e.g. poly([1,2,3],”x”,”coeff”) will input **$1+2x+3x^2$**

**Exception # 1:** Rational Polynomial Functions

Input polynomial functions like how you input it normally:

eg. (x+2) / (x+3)

**Exception # 2**: Irrational Polynomial Functions

Input polynomial fucntions with the standard format of our program but the nth root should be excluded.

eg. Instead of inputting (8 – 2x)^(1/4), just input only (8 – 2x) or poly([8,-2],”x”,”coeff”).

<u>This is because the nth root will be inquired upon by the program after inputting your polynomial function.</u>

eg. After inputting poly([8,-2],”x”,”coeff”), the user will be prompted to input the desired nth root of the radical sign. If user, for example, have inputted nthroot = 4, then it will automatically consider that the entire function is equal to (8 – 2x)^(1/4).

In program console...

*Input your g(x): poly([8,-2],”x”,”coeff”)*

*Input the nthroot: 4*

## IV. Sample Output

INTRODUCTION PAGE

```
"Welcome to Group 6 Iterative Open Method Root Approximator 1.0.1 (G6-IOM 1.0.1)!"
"Version: 1.0.1 (TRIAL VERSION)"
""
"Current Version Capabilities:"
"Fixed-Point Iteration Method: Can approximate up to 25th approximation."
"Newton-Raphson Method: Can approximate up to 20th approximation."
"Secant Method: Can approximate up to 20th approximation."
```

ABORT PAGE (when user inputs “no”)

```
 "Thank you for using our program!"
 "For any inquiries, please contact Head Developer through this email:"
 "bernardoraevon@gmail.com"
-> |
```

SECOND PAGE (when user inputs “yes”)

```
 "Please choose desired program mode:"
 "(a) Fixed-Point Iteration Method"
 "(b) Newton-Raphson Method"
 "(c) Secant Method"
 ""
Your choice:
```

## FIXED-POINT ITERATION METHOD – FUNCTION SELECTION MENU PAGE

```
What type is your g(x)?:

(a) Linear or Nonlinear Polynomial Function

(b) Rational Polynomial Function

(c) Irrational Polynomial Function (Radical)

Your choice:
```

## FIXED-POINT ITERATION METHOD – LINEAR OR NONLINEAR POLYNOMIAL FUNCTION SAMPLE

```
 "Mode: Fixed-Point Iteration Method"

 "Type: Linear or Nonlinear Polynomial Function"

 "How to Use G6-IOM 1.0.1: Input polynomial function in this format: poly([a,b,c,-->nth],%x%

 "Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desire

 ""

 "(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab cc

 ""

 "e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2"

 ""
Input your g(x):  4

Input your x0:  1


 "x0 or initial approximation:  1"

 "x1 or first approximation:  4"

 "x2 or second approximation:  4"

 ""

 "The root is 4"
```

## FIXED-POINT ITERATION METHOD – RATIONAL POLYNOMIAL FUNCTION SAMPLE

```
 "Mode: Fixed-Point Iteration Method"

 "Type: Rational Polynomial Function"

 "How to Use G6-IOM 1.0.1: Input polynomial function like how you normally input rational fu

 "e.g. (x+2)/(x+3)"

 ""
Input your g(x):  (x+2)/(x+3)

Input your x0:  1

 "x0 or initial approximation:  1"

 "x1 or first approximation:  0.75"

 "x2 or second approximation:  0.734"

 "x3 or third approximation:  0.733"

 "x4 or fourth approximation:  0.733"

 ""

 "The root is 0.733"
```

## FIXED-POSITION ITERATION METHOD – IRRATIONAL POLYNOMIAL FUNCTION (RADICAL) SAMPLE

```
 "When iputting your polynomial function, dont mind the nthroot. Wait for program nthroot

 "You are also required to input the nthroot of your irrational polynomial function (radic

 "eg. with poly([1,2,3],%x%,%coeff%), and nthroot input of 3:"

 "This will result in (1+2x+3x^2)^(1/3)"

 ""
Input your g(x):  poly([8,-2],"x","coeff")

Input the nthroot:  4

Input your x0:  1

 "x0 or initial approximation:  1"

 "x1 or first approximation:  1.566"

 "x2 or second approximation:  1.486"

 "x3 or third approximation:  1.498"

 "x4 or fourth approximation:  1.496"

 "x5 or fifth approximation:  1.496"

 ""

 "The root is 1.496"
```

## NEWTON-RAPHSON METHOD SAMPLE

```
"Mode: Newton-Raphson Method"
"How to Use G6-IOM 1.0.1: Input polynomial function in this format: poly([a,b,c,-->nth],%
"Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desi
""
"(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab
""
"e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2"
""
nput your f(x):  poly([-8,2,0,0,1],"x","coeff")
nput derivative of f(x):  poly([2,0,0,4],"x","coeff")
nput your x0:  1

"x0 or initial approximation:  1"
"f(x0):  -5"
"first derivative of f(x0):  6"
""
"x1 or first approximation:  1.834"
"f(x1):  6.982"
"first derivative of f(x1):  26.676"
""
"x2 or second approximation:  1.572"
"f(x2):  1.251"
"first derivative of f(x2):  17.539"
""
"x3 or third approximation:  1.501"
"f(x3):  0.079"
"first derivative of f(x3):  15.528"
""
"x4 or fourth approximation:  1.496"
"f(x4):  0.001"
"first derivative of f(x4):  15.393"
""
"x5 or fifth approximation:  1.496"
""
"The root is 1.496"
```

## SECANT METHOD SAMPLE

```
 "Mode: Secant Method"
 "How to Use G6-IOM 1.0.1: Input polynomial function in this format: poly([a,b,c,-->nth],%
 "Wherein: Letters a,b,c,-->nth (excluding x) corresponds to the coefficients of your desi
 ""
 "(IMPORTANT): The symbol % should be replaced with quotation mark (This is due to Scilab
 ""
 "e.g. poly([1,2,3],%x%,%coeff%) will input 1+2x+3x^2"
 ""
Input your f(x):  poly([-8,2,0,0,1],"x","coeff")
Input your x0:  1
Input your x1:  2

 ""
 "Iteration 1:"
 "x0 or initial approximation:  1"
 "x1 or first approximation:  2"
 "f(x0):  -5"
 "f(x1):  12"
 ""
 "x2 or second approximation:  1.295"
 ""
 "Iteration 2:"
 "x1 or first approximation:  2"
 "x2 or second approximation:  1.295"
 "f(x1):  12"
 "f(x2):  -2.597"
 ""
 "x3 or third approximation:  1.421"
 ""
 "Iteration 3:"
 "x2 or second approximation:  1.295"
 "x3 or third approximation:  1.421"
 "f(x2):  -2.597"
 "f(x3):  -1.08"
 ""
```

```
""
"Iteration 6:"
"x5 or fifth approximation:  1.495"
"x6 or sixth approximation:  1.496"
"f(x5):  -0.014"
"f(x6):  0.001"
""
"x7 or seventh approximation:  1.496"
""
"The root is 1.496"
```

**The console screen appears much more cleaner when using Scilab 6.0.2 rather than the current version of Scilab. In Scilab 6.1.1, quotation marks are displayed which makes it look messier whereas compared to Scilab 6.0.2, quotation marks are not displayed on the console screen.**

**– Head Developer's Remark**

## V. Program Accuracy

FIXED-POINT ITERATION METHOD

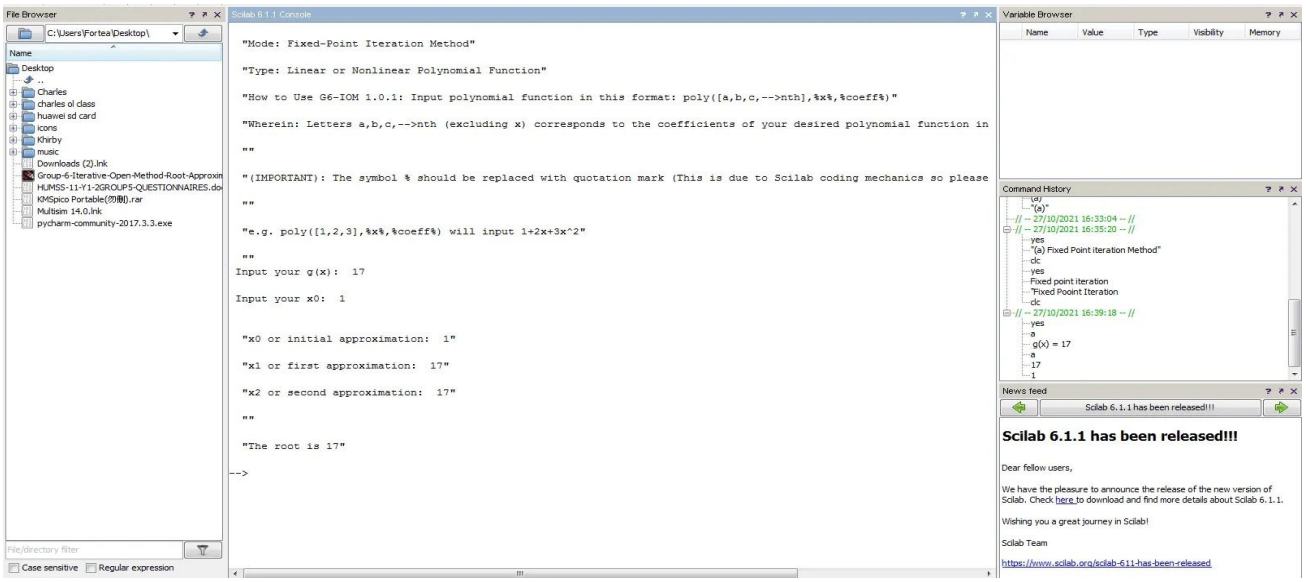
- **Linear or Nonlinear Polynomial Function**

<u>Accuracy Test # 1</u>

- g(x) =17
- x0 =1

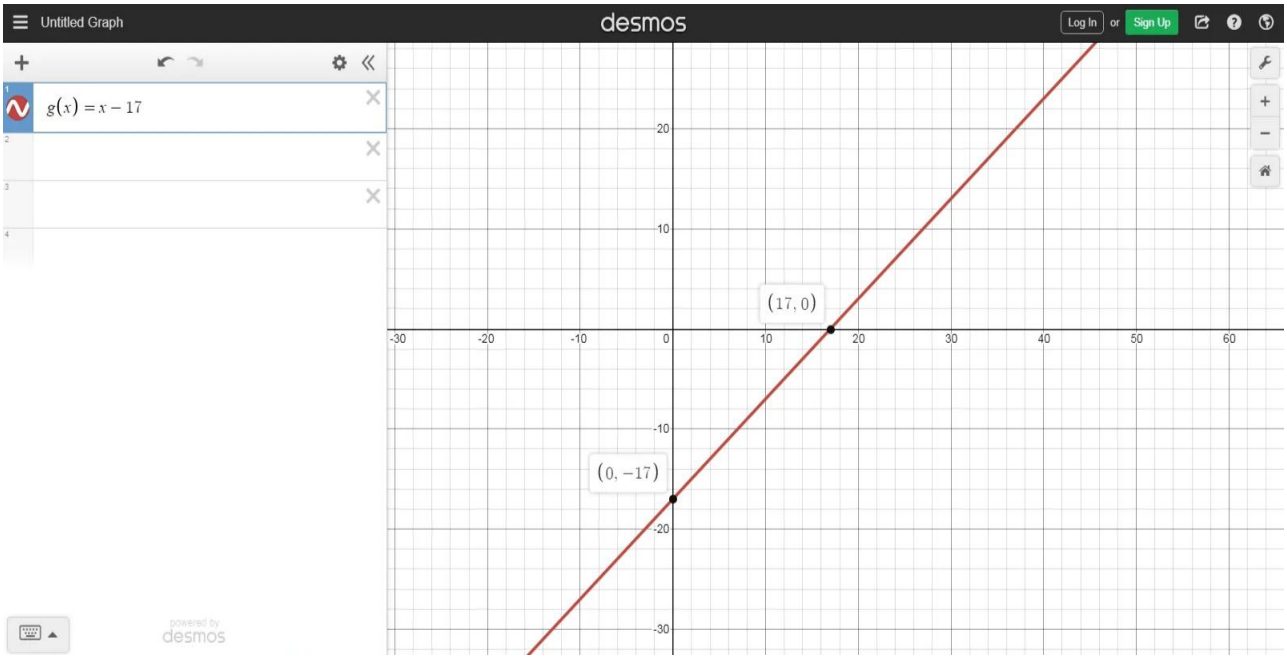
**G6–IOMRA 1.0.1 Results:**

Root = 17




**Desmos Graphing Calculator Results:**

Root = 17



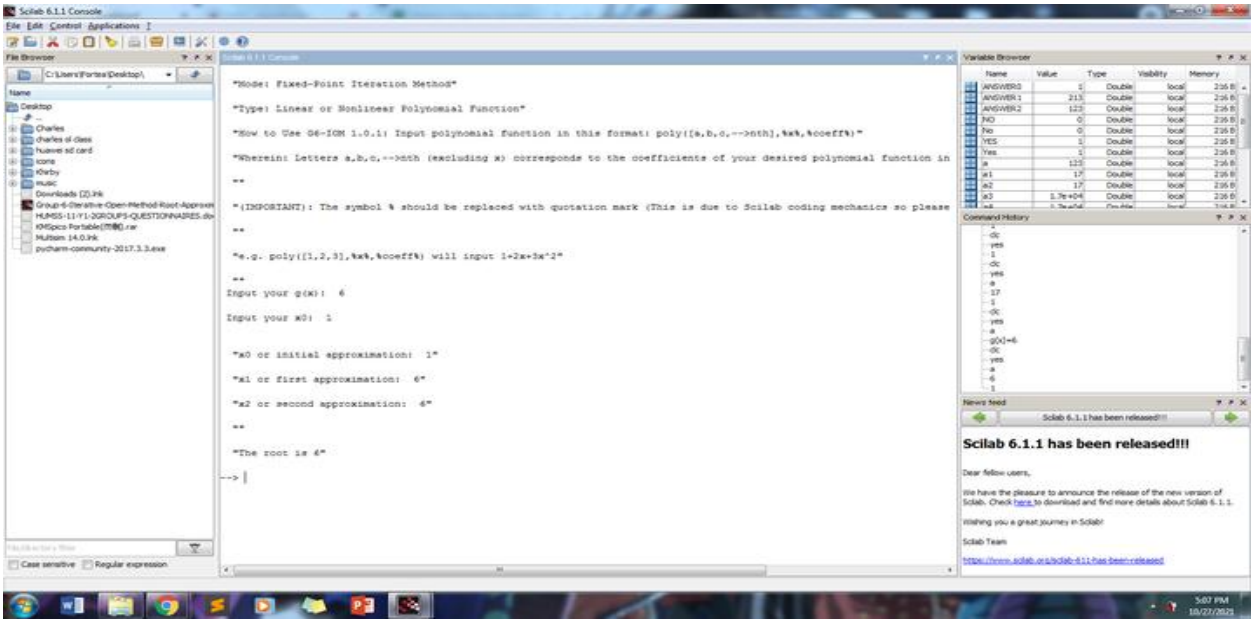
FIXED-POINT ITERATION METHOD

- **Linear or Nonlinear Polynomial Function**

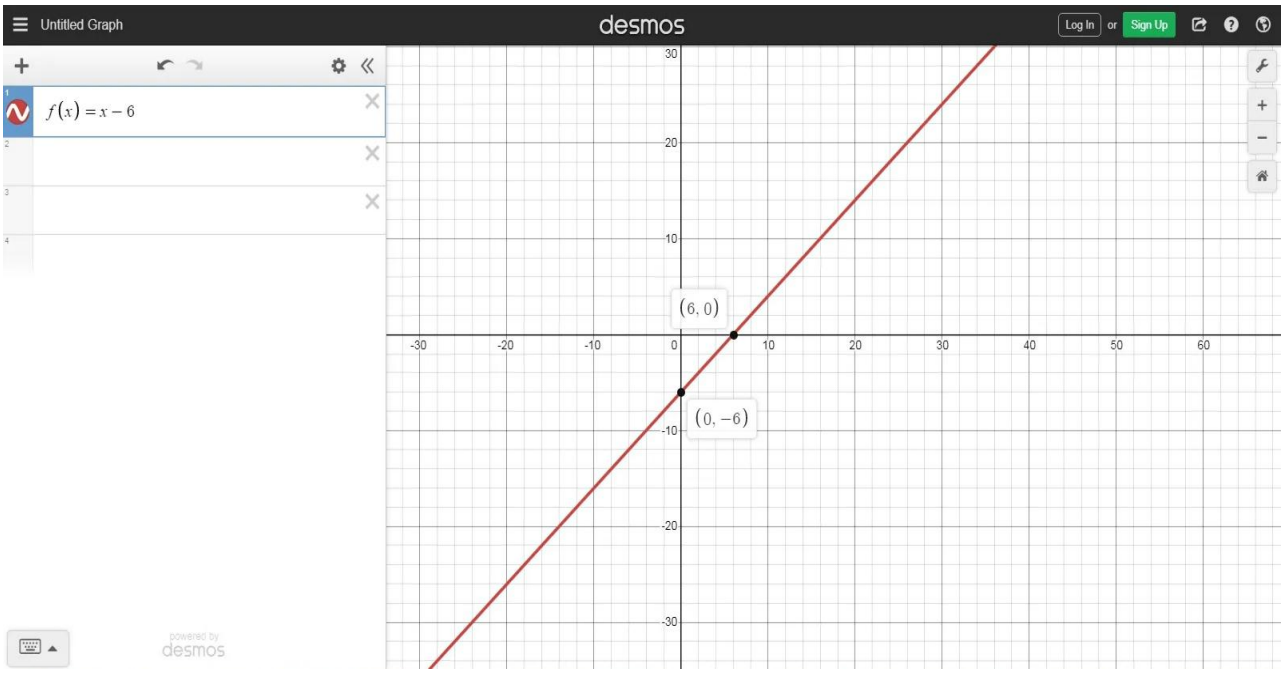
Accuracy Test # 2

- g(x) =6
- x0 =1

**G6–IOMRA 1.0.1 Results:**
Root = 6




**Desmos Graphing Calculator Results:**
Root = 6



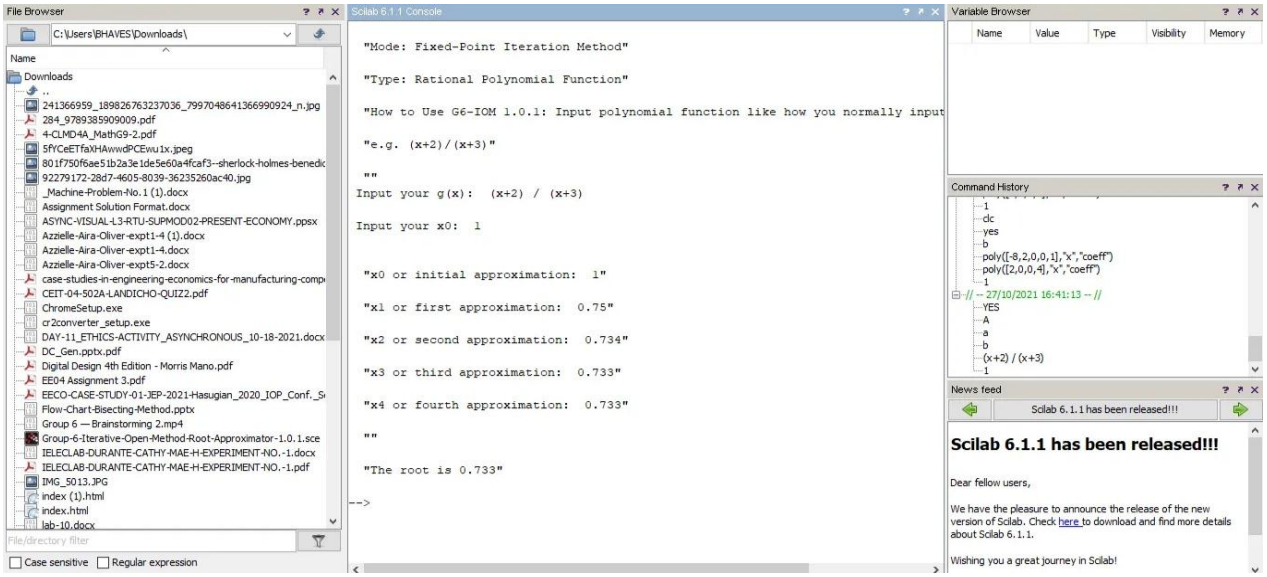
FIXED-POINT ITERATION METHOD

- **Rational Polynomial Function**

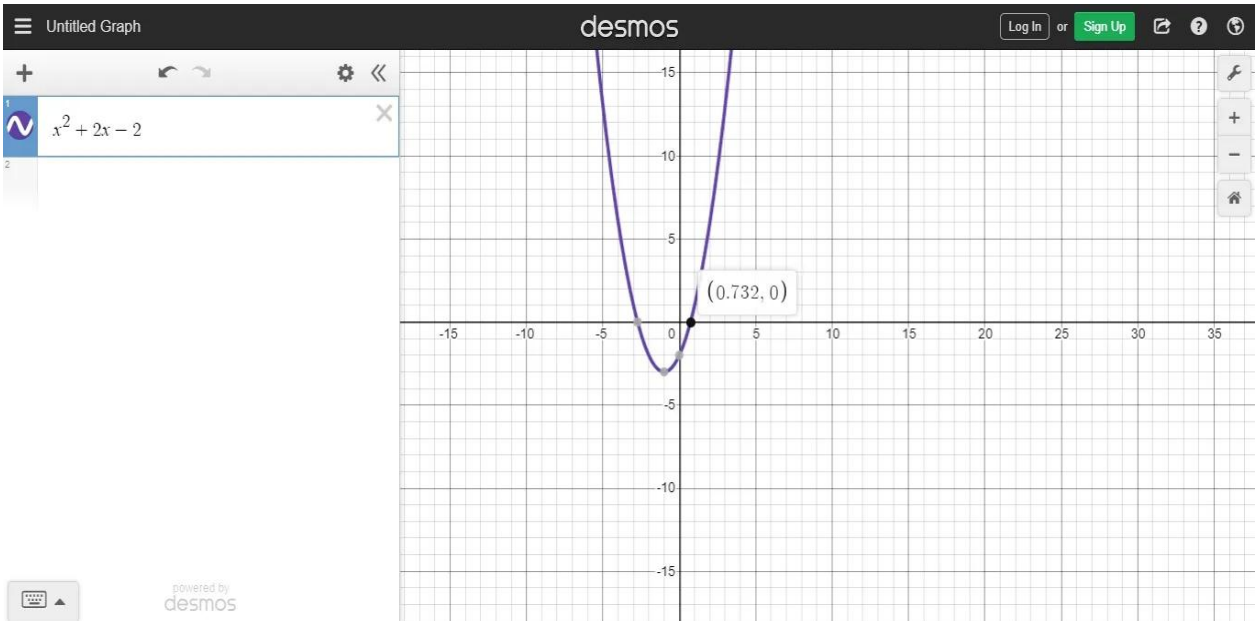
Accuracy Test # 1

- g(x) = (x+2) / (x+3)
- x0 = 1

**G6–IOMRA 1.0.1 Results:**
Root = 0.733




**Desmos Graphing Calculator Results:**
Root = 0.732



FIXED-POINT ITERATION METHOD

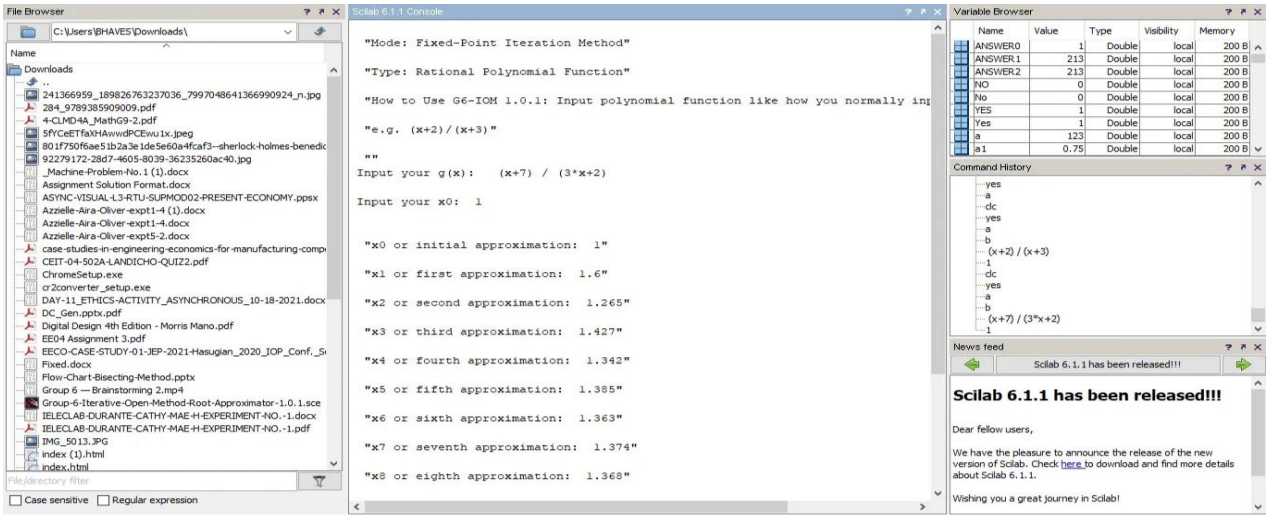
- **Rational Polynomial Function**

Accuracy Test # 2

- g(x) = (x+7) /(3*x+2)
- x0 = 1

## G6–IOMRA 1.0.1 Results:

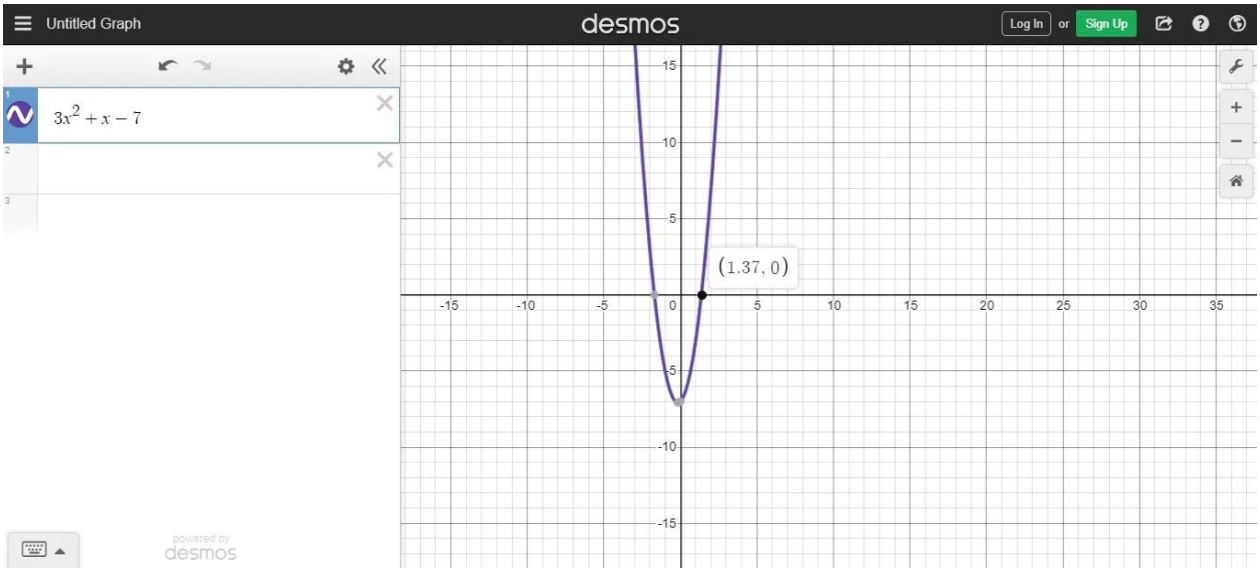
Root = 1.37




```
"x9 or ninth approximation:  1.371"
"x10 or tenth approximation:  1.37"
"x11 or eleventh approximation:  1.37"
""
"The root is 1.37"
```

## Desmos Graphing Calculator Results:

Root = 1.37



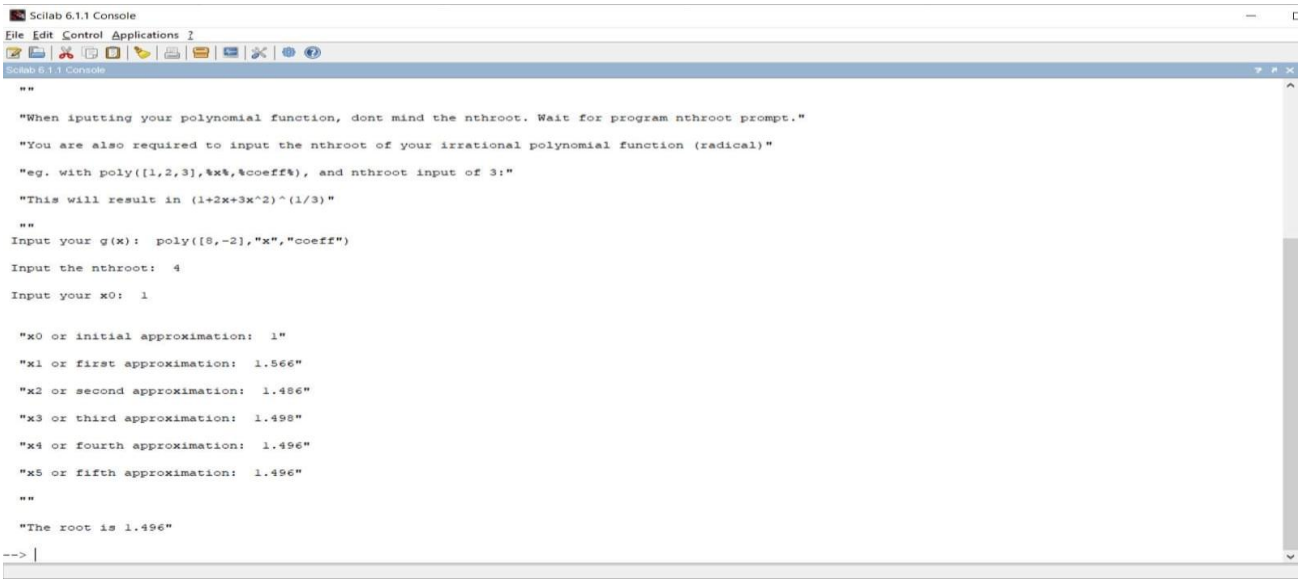
FIXED-POINT ITERATION METHOD

- **Irrational Polynomial Function (Radical)**

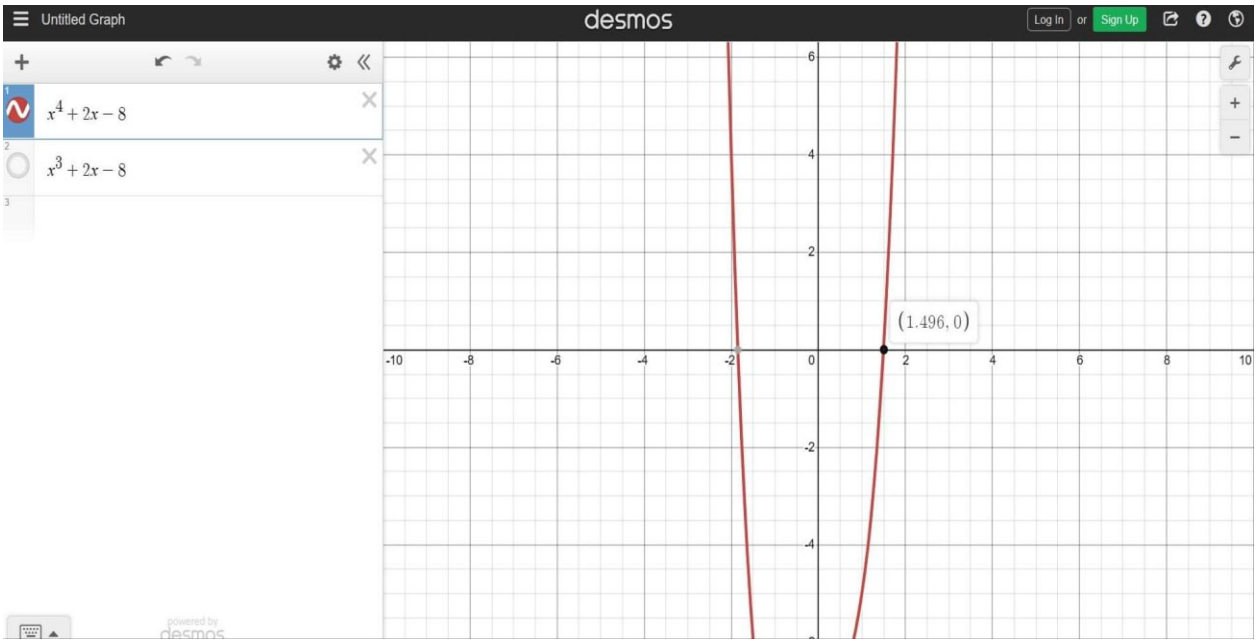
Accuracy Test # 1

- g(x) = poly([8,-2],”x”,”coeff”) or (8 – 2x)
- nthroot = 4
- x0 = 1

**G6–IOMRA 1.0.1 Results:**
Root = 1.496




**Desmos Graphing Calculator Results:**
Root = 1.496



FIXED-POINT ITERATION METHOD

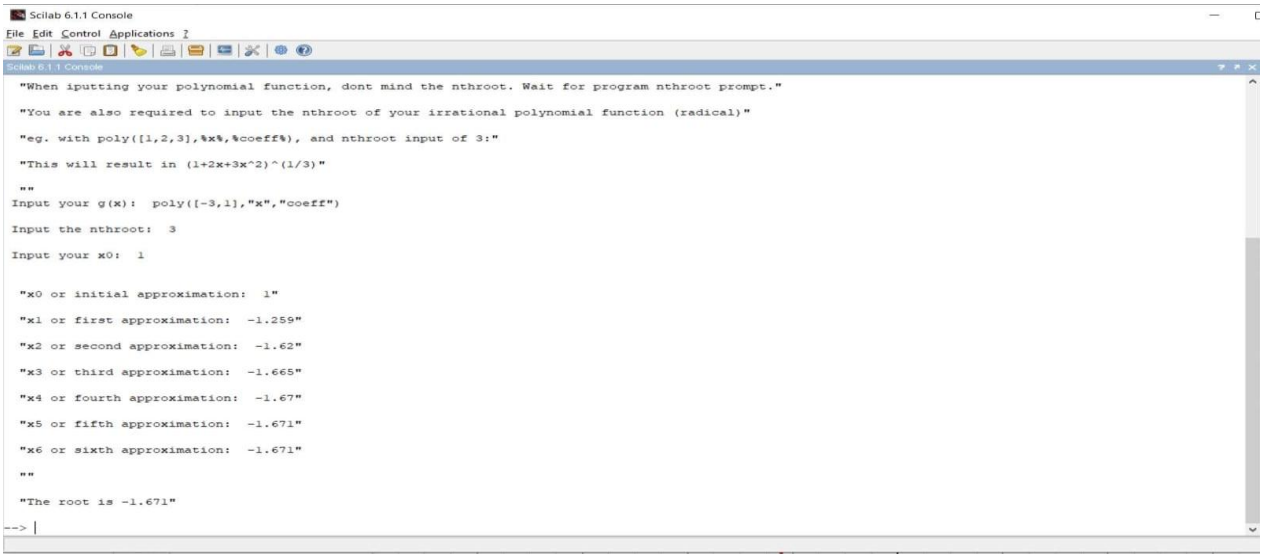
- **Irrational Polynomial Function (Radical)**

Accuracy Test # 2

- g(x) = poly([-3,1],"x","coeff") or (-3 + x)
- nthroot = 3
- x0 = 1

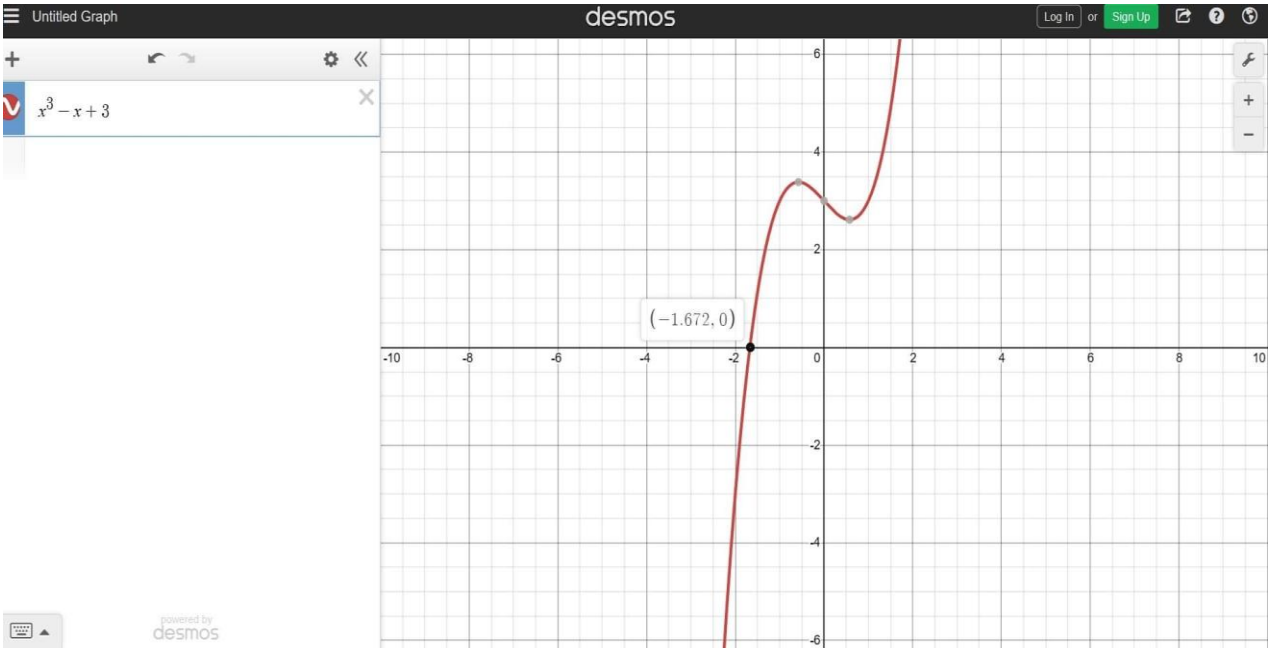
**G6–IOMRA 1.0.1 Results:**

Root = – 1.671




**Desmos Graphing Calculator Results:**

Root = – 1.672



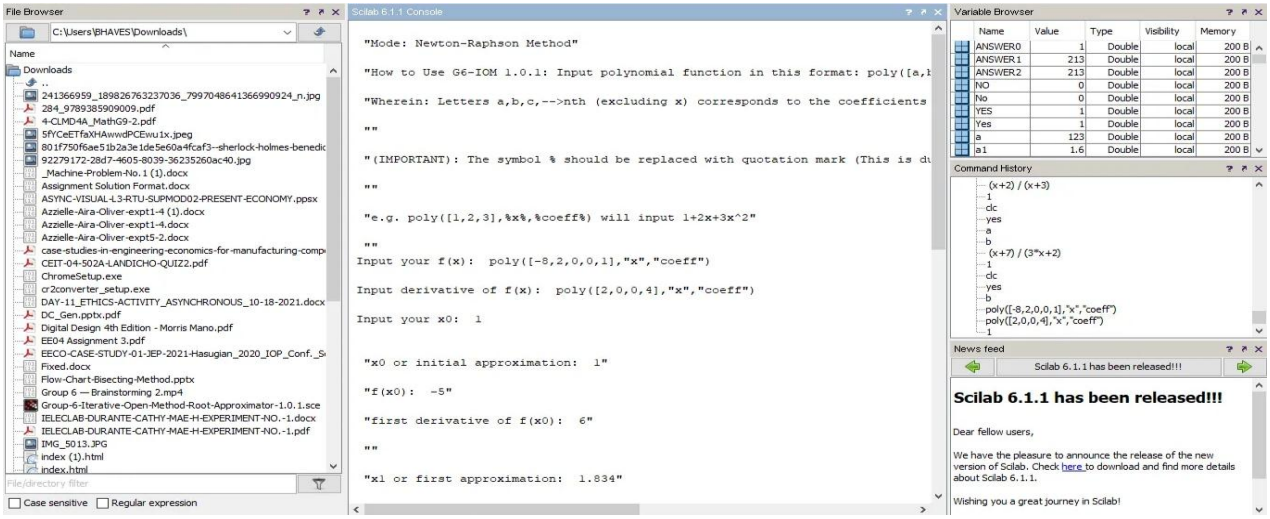
NEWTON-RAPHSON METHOD

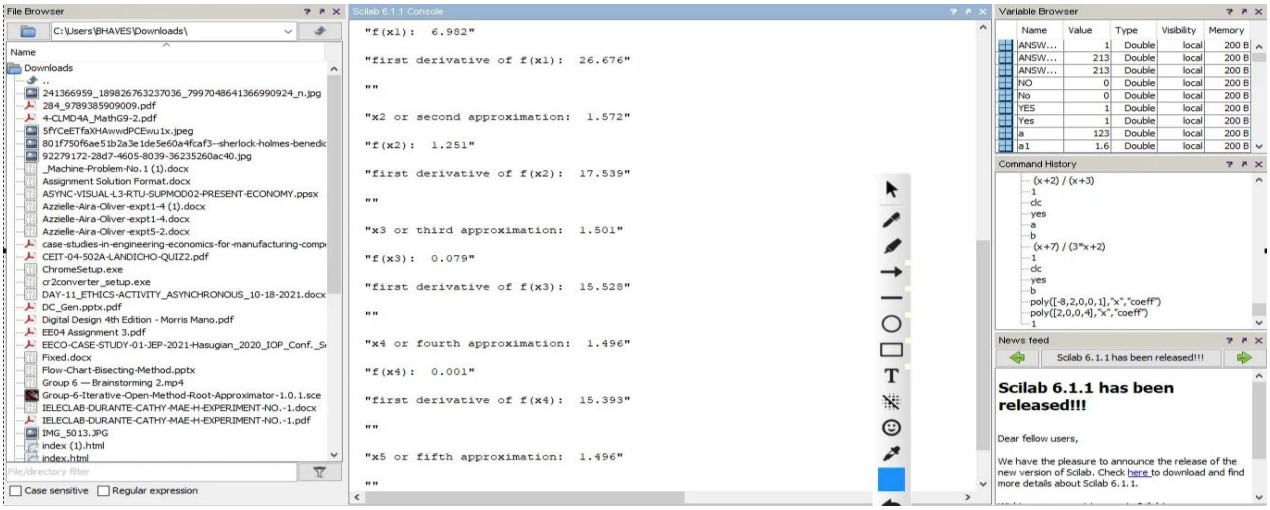
Accuracy Test # 1

- f(x) = poly([-8,2,0,0,1],"x","coeff") or $x^4 + 2x - 8$
- f'(x) = poly([2,0,0,4],"x","coeff") or $4x^3 + 2$
- x0 = 1

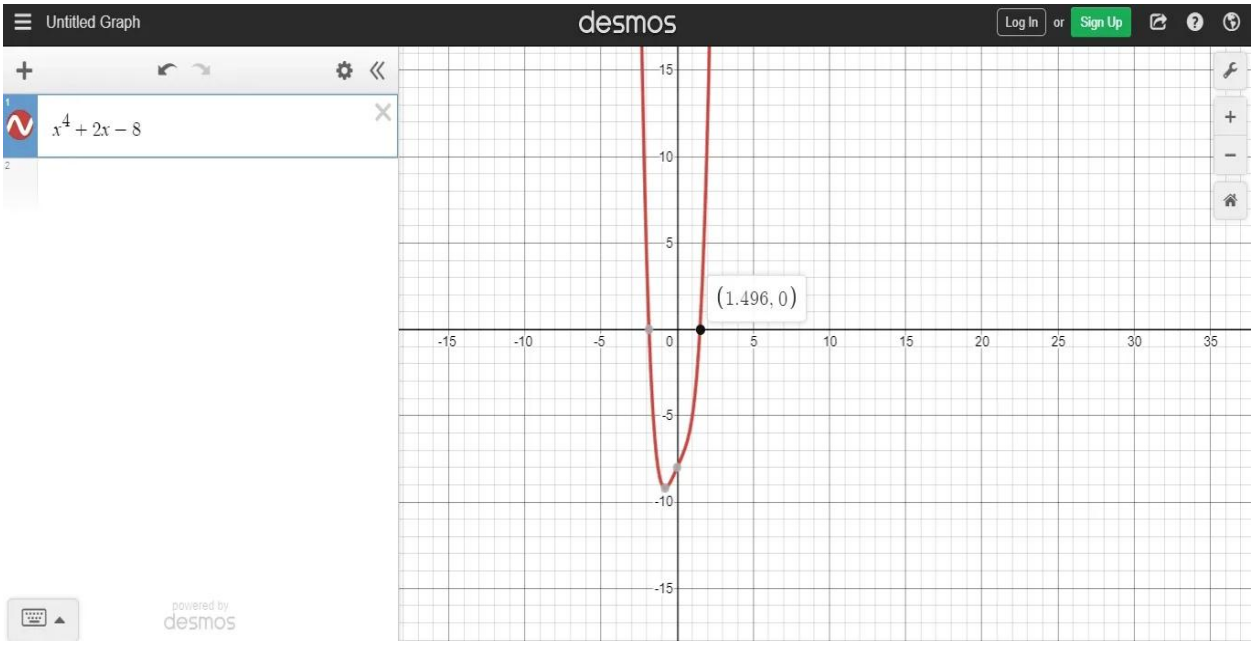
**G6–IOMRA 1.0.1 Results:**

Root = 1.496




**Desmos Graphing Calculator Results:**
Root = 1.496

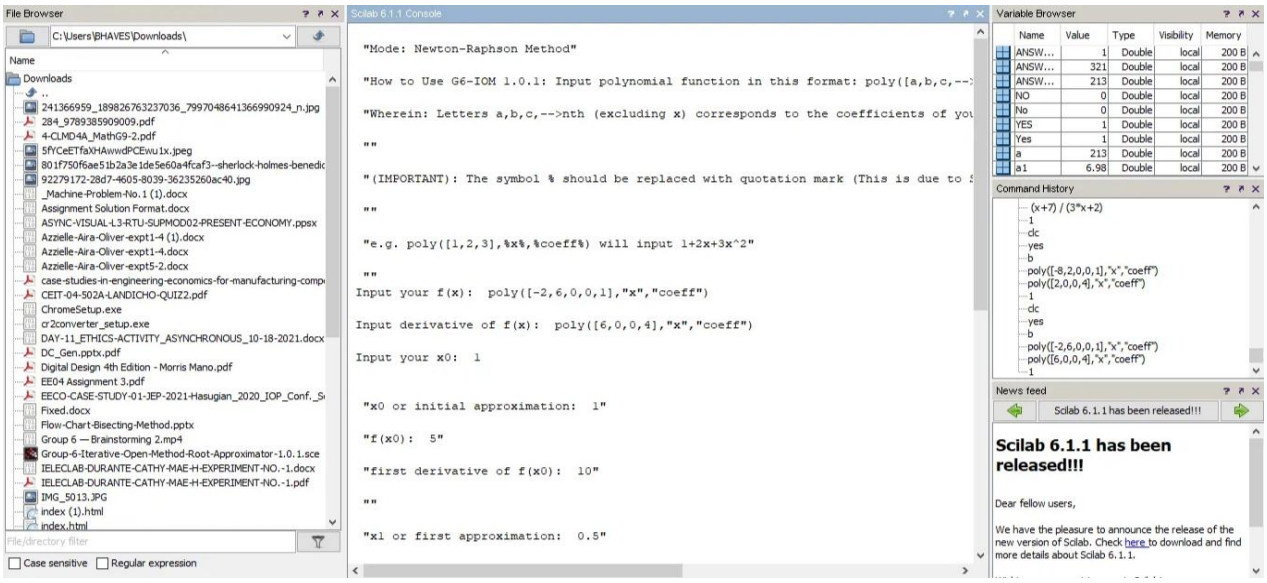


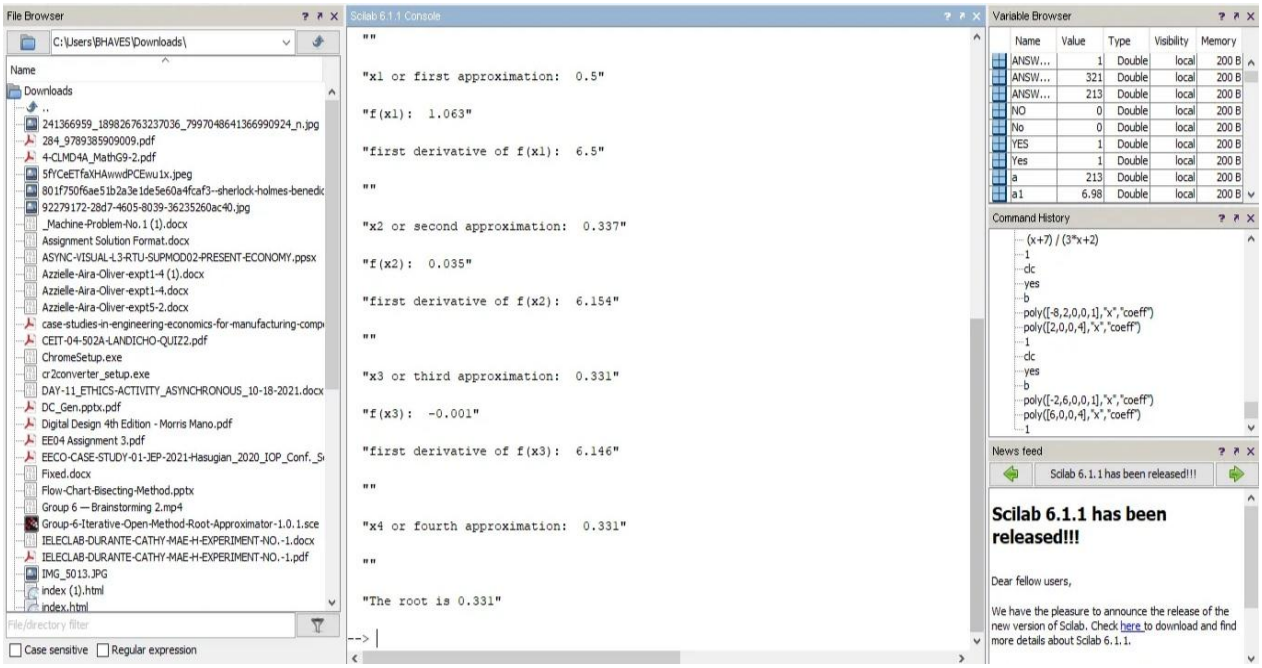

NEWTON-RAPHSON METHOD

Accuracy Test # 2

- f(x) = poly([-2,6,0,0,1],"x","coeff") or $x^4 + 6x - 2$
- f'(x) = poly([6,0,0,4],"x","coeff") or $4x^3 + 6$
- x0 = 1

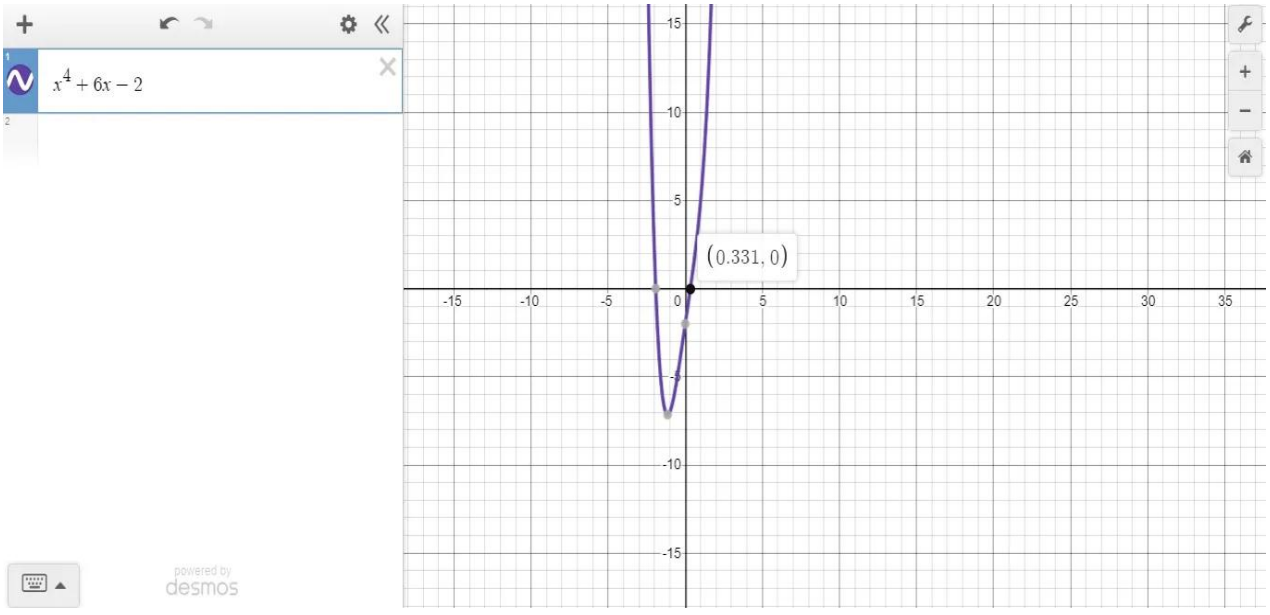
**G6–IOMRA 1.0.1 Results:**
Root = 0.331






**Desmos Graphing Calculator Results:**

Root = 0.331



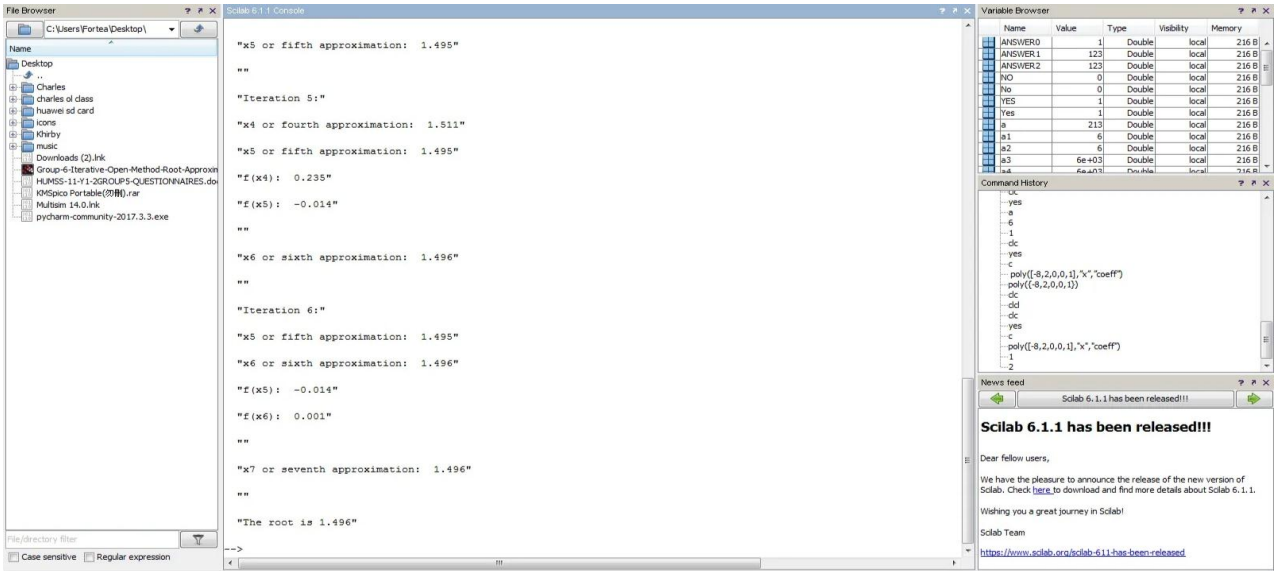
SECANT METHOD

Accuracy Test # 1

- f(x) = poly([-8,2,0,0,1],"x","coeff") or $x^4 + 2x - 8$
- x0 = 1
- x1 = 2

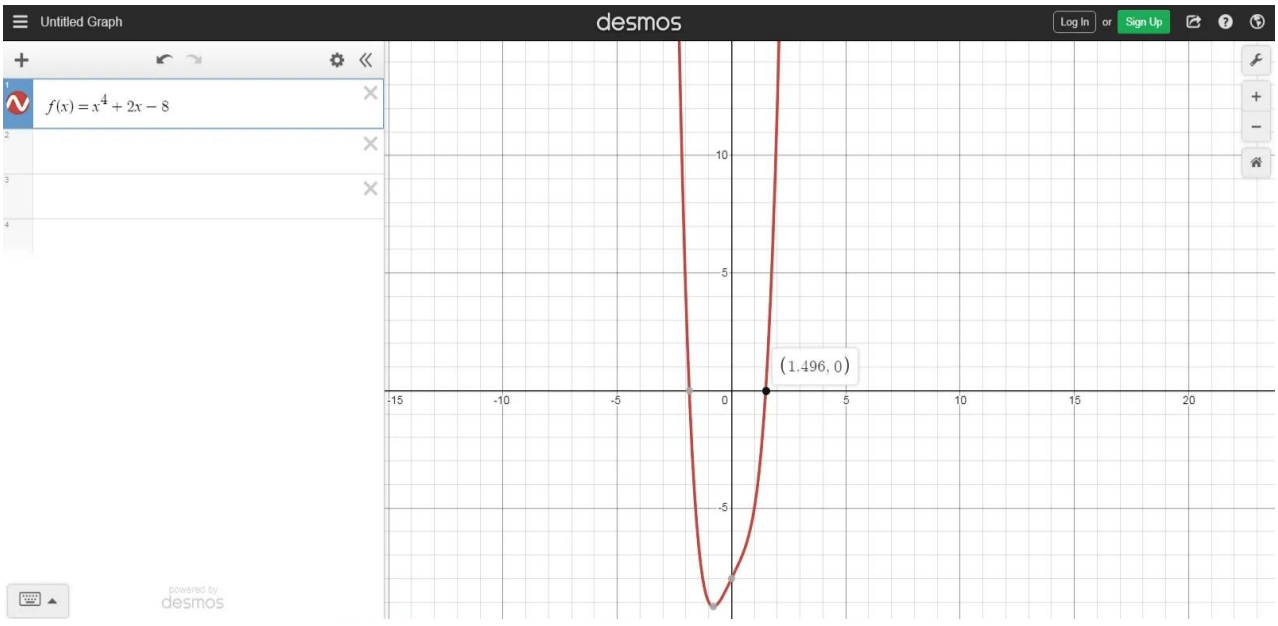
**G6–IOMRA 1.0.1 Results:**

Root = 1.496




**Desmos Graphing Calculator Results:**

Root = 1.496



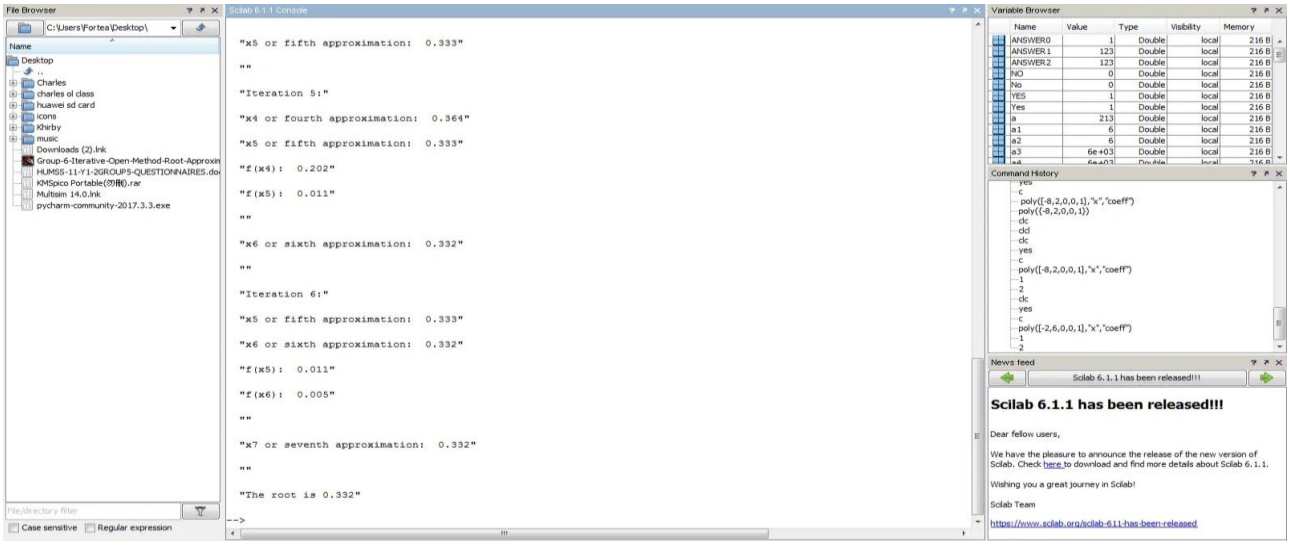
SECANT METHOD

Accuracy Test # 2

- f(x) = poly([-2,6,0,0,1],"x","coeff") or $x^4 + 6x - 2$
- x0 = 1
- x1 = 2

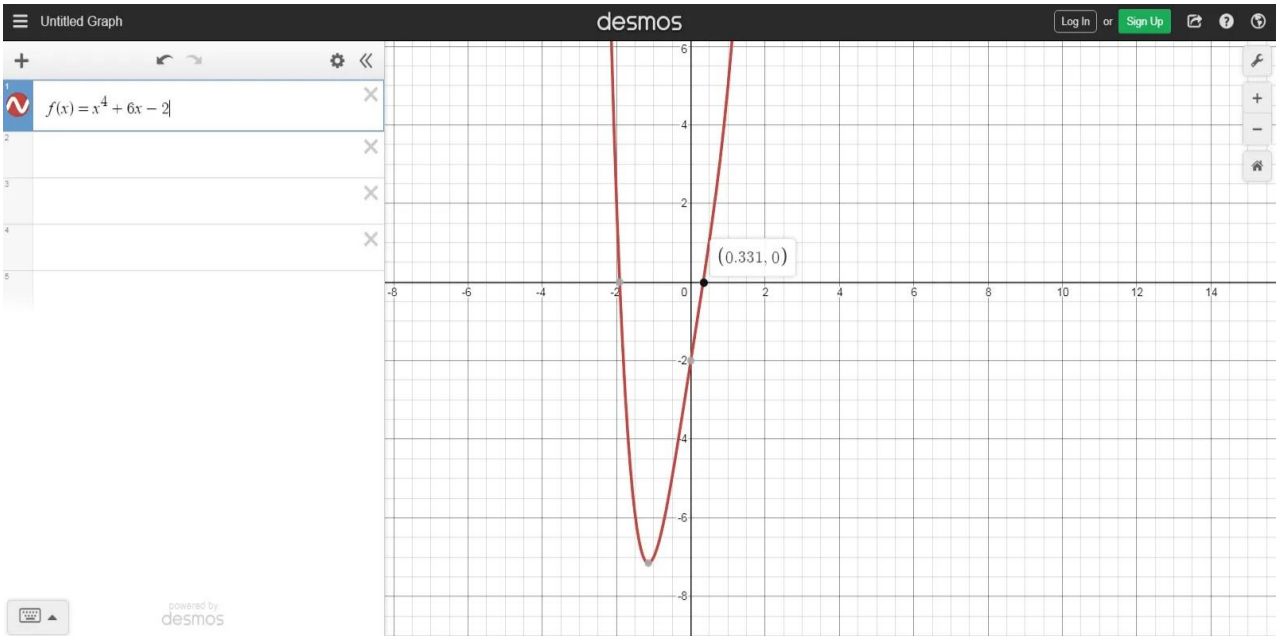
**G6–IOMRA 1.0.1 Results:**

Root = 0.332




**Desmos Graphing Calculator Results:**

Root = 0.331



## VI. Development Team Contributions

Development Team Members:

- ❖ Bernardo, Raevon Thaddeus C.
- Head Developer & Programmer
- Designed the algorithms of the working program
- Final debugger of the program

- ❖ Bertumen, Charles Jefferson
- Assistant Developer & Programmer
- Assisted in conceptualizing the algorithms of the program
- Assisted in assessing the performance of the trial version
- Assisted in debugging the program

- ❖ Cabanes, Christine Joy P.
- Assistant Developer & Programmer
- Assisted in conceptualizing the algorithms of the program
- Assisted in assessing the performance of the trial version
- Assisted in debugging the program

- ❖ Cesar, John Lester M.
- Assistant Developer & Programmer
- Assisted in conceptualizing the algorithms of the program
- Assisted in assessing the performance of the trial version
- Assisted in debugging the program

- ❖ Landicho, Bhaves Nicolette D.
- Assistant Developer & Programmer
- Assisted in conceptualizing the algorithms of the program
- Assisted in assessing the performance of the trial version
- Assisted in debugging the program

- ❖ Solis, Johnloyd P.
- Assistant Developer & Programmer
- Assisted in conceptualizing the algorithms of the program
- Assisted in assessing the performance of the trial version
- Assisted in debugging the program

**The development of the program was conducted systematically in order to maximize work efficiency, therefore, the final output was the result of total team effort and cooperation.**

**Head Developer's Remark**